"Porque la arquitectura es el arte que más se esfuerza por reproducir en su ritmo el orden del universo, que los antiguos llamaban *kosmos*, es decir, adorno, pues es como un gran animal en el que resplandece la perfección y proporción de todos sus miembros."

Adso en *El nombre de la rosa*,
de Umberto Eco

Esta tesis está dedicada a
quienes me han enseñado a navegar,
quienes han construido mi barco,
y quienes hoy soplan en sus velas.

Agradecimientos

Un trabajo de investigación de la envergadura de una tesis doctoral no puede llegar a buen puerto sin la colaboración de muchas personas. Quisiera agradecer en mi primer lugar a mi familia el apoyo incondicional que siempre me han prestado en todos los sentidos.

Merecen también una mención especial mis directores, el Dr. Ramón Doallo y el Dr. Emilio L. Zapata, por la gran cantidad de horas que han dedicado a la dirección de este trabajo, y la orientación que me proporcionaron en las encrucijadas que se me presentaron.

Ha habido también mucha gente que me ha ayudado en multitud de problemas técnicos, desde la configuración de mi estación de trabajo, en la que la colaboración de Miguel Alonso fue fundamental, hasta el mantenimiento de la red y muchos de los computadores que he usado, realizados hoy por hoy en parte por el personal del Centro de Cálculo da Facultade de Informática (CeCaFI).

Quisiera agradecer a mis compañeros en la docencia de la asignatura que he venido impartiendo los cuatro últimos años, Tecnoloxía de Computadores, en especial José R. Sanjurjo y Carlos Vázquez, por las numerosas ocasiones en que han cargado sobre sus hombros con la mayor parte del peso de la misma.

Entre las personas que me han aguantado estoicamente a la hora del café quisiera destacar el valor de Cristina Mejuto y Margarita Amor, debiéndole también a esta última la consecución de un estilo LaTeX para la escritura de esta memoria.

En lo tocante a los organismos que han hecho posible la realización de esta tesis, el Departamento de Electrónica e Sistemas da Universidade da Coruña puso los medios necesarios para ello, apoyado por la Universidad en la financiación de los desplazamientos a los congresos relacionados con este trabajo. Otra parte esencial de la financiación la supusieron los proyec-

tos TIC96-1125-C03 del CICYT y XUGA20605B96 de la Xunta de Galicia. También el Edinburgh Parallel Computer Centre (EPCC) posibilitó, mediante la concesión de una estancia a través del programa TRACS de la Unión Europea, la preparación y ejecución de varias simulaciones en su CRAY T3D.

Por último, quisiera agradecer a toda la comunidad investigadora en el área de arquitectura de computadores los avances producidos en los mismos. Hubiera sido muy difícil llevar a cabo muchas de las pruebas necesarias para verificar nuestros modelos con los computadores de que disponíamos al comenzar esta tesis.

Índice general

1. El análisis de la jerarquía de memoria **1**

1.1. Estrategias para el estudio del comportamiento de la caché . . 2

1.2. Trabajos previos 4

1.3. Objetivos y organización de la tesis 6

2. Aspectos generales del modelado **9**

2.1. Parámetros de entrada 9

2.2. Ideas básicas 10

2.2.1. Unión de vectores de área 11

2.2.2. Número de líneas en un vector compitiendo por el mismo conjunto de la caché 12

2.3. Patrones de acceso 13

2.3.1. Acceso secuencial 14

2.3.2. Acceso a líneas con una probabilidad uniforme de referencia 15

2.3.3. Acceso a grupos de elementos con una probabilidad uniforme de referencia 17

2.3.4. Acceso a áreas desplazadas con referencias sucesivas . . 19

2.3.5. Acceso a regiones separadas por una distancia constante 22

2.4. Simplificaciones en el modelado de cachés de correspondencia directa 28

2.4.1. Patrones de acceso 29

3. Modelado de cachés asociativas por conjuntos para computaciones irregulares 31

3.1. Un procedimiento sistemático 33

3.2. Producto matriz dispersa-vector 34

3.2.1. Vectores `A` y `C` 35

3.2.2. Vectores `R` y `D` 36

3.2.3. Vector `X` 36

3.2.4. Validación y análisis 38

3.3. Producto matriz dispersa-matriz densa: orden JIK 42

3.3.1. Vectores `A` y `C` 43

3.3.2. Vector `R` 44

3.3.3. Validación 45

3.4. Producto matriz dispersa-matriz densa: orden IKJ 46

3.4.1. Vector `R` 46

3.4.2. Vectores `A` y `C` 47

3.4.3. Matriz `D` 47

3.4.4. Matriz `B` 48

3.4.5. Validación 49

3.5. Producto matriz dispersa-matriz densa: orden IJK 49

3.5.1. Vectores `A` y `C` 51

3.5.2. Matriz `D` 51

3.5.3. Matriz `B` 52

3.5.4. Validación y análisis 53

3.6. Trasposición de una matriz dispersa 56

3.6.1. Vector `A` 57

3.6.2. Vector `R` 58

3.6.3. Vectores `AT` y `CT` 58

3.6.4. Vector `C` 60

3.6.5. Vector `RT` 62

3.6.6. Validación y análisis 65

3.7. Producto matriz dispersa-matriz densa con orden IKJ altamente optimizado 67
3.7.1. Vector `R` 69
3.7.2. Vector `R2` 74
3.7.3. Vector `A` 74
3.7.4. Vector `C` 76
3.7.5. Vector `D` 77
3.7.6. Vector `B` 79
3.7.7. Vector `WB` 79
3.7.8. Validación y análisis 83
3.8. Tiempos de simulación versus tiempos de modelado 90

4. Otras distribuciones: matrices banda 93
4.1. Bandas uniformes 93
4.1.1. Producto matriz dispersa-vector 94
4.1.2. Producto matriz dispersa-matriz densa: orden JIK . . . 97
4.1.3. Producto matriz dispersa-matriz densa: orden IKJ . . . 98
4.1.4. Producto matriz dispersa-matriz densa: orden IJK . . . 100
4.1.5. Trasposición de una matriz dispersa 103
4.2. Bandas no uniformes 112
4.2.1. Producto matriz dispersa-vector 113
4.2.2. Producto matriz dispersa-matriz densa: orden JIK . . . 117
4.2.3. Producto matriz dispersa-matriz densa: orden IKJ . . . 118
4.2.4. Producto matriz dispersa-matriz densa: orden IJK . . . 122
4.2.5. Trasposición de una matriz dispersa 123

5. Una aproximación al modelado automático para patrones de acceso regulares 129
5.1. Ámbito inicial del modelado 130
5.1.1. Anidamientos perfectos de bucles 131
5.1.2. Ecuaciones del número de fallos 131

5.1.3. Cálculo de los vectores de área de interferencia 133
5.1.4. Posiciones relativas de las estructuras de datos 135
5.1.5. Referencias secuenciales en traslación 136
5.1.6. Blocking 138
5.2. Referencias múltiples a una estructura de datos 140
5.2.1. Formulación del número de fallos en la dimensión donde difieren las referencias 140
5.2.2. Referencias que difieren en varias dimensiones 143
5.2.3. Cálculo de vectores de área 143
5.3. Anidamientos no perfectos y reuso de datos entre bucles . . . 146
5.4. Verificación del modelado automático 149
5.5. Tiempos de simulación versus tiempos de modelado 157

Conclusiones y principales aportaciones **163**

Bibliografía **166**

Índice de cuadros

2.1. Notación empleada. 10

3.1. Parámetros de la matriz. 32

3.2. Desviación del modelo para el producto matriz dispersa-vector. 39

3.3. Desviación del modelo para el producto matriz dispersa-matriz densa con orden JIK. 45

3.4. Desviación del modelo para el producto matriz dispersa-matriz densa con orden IKJ. 50

3.5. Desviación del modelo para el producto matriz dispersa-matriz densa con orden IJK. 53

3.6. Desviación del modelo para la trasposición de una matriz dispersa. 65

3.7. Desviación del modelo para el producto matriz dispersa-matriz densa optimizado. 83

3.8. Desviación del tiempo de ejecución usando el bloque propuesto por el modelo como porcentaje del tiempo óptimo de ejecución. M y N en miles, $H = 1000$. 85

3.9. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz dispersa-matriz densa. 91

3.10. Tiempos de usuario de la simulación mediante dineroIII, el simulador simplificado y tiempos de ejecución del modelo para la trasposición de una matriz dispersa. 92

3.11. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz dispersa-matriz densa IKJ altamente optimizado. 92

4.1. Desviación del modelo para el producto matriz dispersa banda-vector. 95
4.2. Desviación del modelo para el producto matriz dispersa banda-matriz densa con orden JIK. 98
4.3. Desviación del modelo para el producto matriz dispersa banda-matriz densa con orden IKJ. 101
4.4. Desviación del modelo para el producto matriz dispersa banda-matriz densa con orden IJK. 103
4.5. Desviación del modelo para la trasposición de una matriz dispersa en banda. 108
4.6. Desviación del modelo para el producto matriz dispersa banda no uniforme-vector. 116
4.7. Desviación del modelo para el producto matriz dispersa banda no uniforme-matriz densa con orden JIK. 117
4.8. Desviación del modelo para el producto matriz dispersa banda no uniforme-matriz densa con orden IKJ. 121
4.9. Desviación del modelo para el producto matriz dispersa banda no uniforme-matriz densa con orden IJK. 124
4.10. Desviación del modelo para la trasposición de una matriz banda no uniforme. 128

5.1. Datos de validación del modelo automatizado para el producto matriz densa-matriz densa con orden JIK. 155
5.2. Datos de validación del modelo automatizado para el código Stencil. 155
5.3. Datos de validación del modelo automatizado para el método de Jacobi en dos dimensiones. 156
5.4. Datos de validación del modelo automatizado para el producto matriz densa-matriz densa con orden IKJ y *blocking*. 157
5.5. Datos de validación del modelo automatizado para el producto matriz densa-matriz densa con orden IKJ, *blocking* y copia con trasposición del bloque. 158
5.6. Datos de validación del modelo automatizado para el código de resolución de ecuaciones diferenciales mediante el método de Gauss-Seidel. 158

5.7. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz densa-matriz densa con orden JIK. 159

5.8. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos del código de Stencil. 159

5.9. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos del código de Jacobi bidimensional. . . 160

5.10. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz densa-matriz densa con orden IKJ y *blocking*. 160

5.11. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz densa-matriz densa con orden IKJ, *blocking* y copia con trasposición del bloque. . 161

5.12. Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos del código de resolución de ecuaciones diferenciales parciales mediante el método de Gauss-Seidel. . . 161

Índice de figuras

2.1. Vectores de área correspondientes a accesos un vector `U` y un vector `V` en una caché con $N_{\mathrm{k}} = 4$ y $K = 2$ y vector de área total resultante. 12

2.2. Código constituido únicamente por accesos secuenciales. . . . 14

2.3. Vectores de área correspondientes a un acceso secuencial y a un acceso a líneas con una probabilidad uniforme de referencia p de un vector que cubre 11 líneas de una caché con $N_{\mathrm{k}} = 8$ y $K = 2$. 15

2.4. Código con un acceso con una probabilidad uniforme de referencia por línea para el vector `X`. 15

2.5. Código con un acceso a grupos de elementos con una probabilidad uniforme de referencia para el vector `X`. 17

2.6. Correspondencia entre la localización de las entradas en una matriz banda a trasponer y los grupos a donde son copiadas en los vectores que definen la matriz traspuesta. 19

2.7. Código con accesos a regiones separadas por una distancia constante. 23

2.8. Orden de acceso de las regiones que finalizan en las posiciones i y j en función de la relación entre `T`(i) y `T`(j). 26

3.1. Producto matriz dispersa-vector. 34

3.2. Número de fallos en una caché de 2K palabras asociativa por conjuntos de 4 vías durante el producto matriz dispersa-vector de una matriz $10K \times 10K$ en función de L_{s} y p_n. 40

3.3. Número de fallos durante el producto matriz dispersa-vector de una matriz $10K \times 10K$ con $p_n = 0{,}18$ en una caché de 2K palabras en función de L_s para varios grados de asociatividad. 41

3.4. Número de fallos de cada vector durante el producto matriz dispersa-vector de una matriz $10K \times 10K$ con $p_n = 0{,}05$ en una caché de dos vías con $L_s = 8$ palabras en función de C_s. . 41

3.5. Producto matriz dispersa-matriz densa con orden JIK. 43

3.6. Producto matriz dispersa-matriz densa con orden IKJ. 46

3.7. Producto matriz dispersa-matriz densa con orden IJK. 50

3.8. Número de fallos durante el producto matriz dispersa-matriz densa en función del ordenamiento, el tamaño de la caché y el grado de asociatividad para una matriz dispersa 10Kx10K con $p_n = 0{,}05$ y $H = 1K$ usando $L_s = 8$. 54

3.9. Número de fallos durante el producto matriz dispersa-matriz densa en función del ordenamiento, el tamaño de la línea de caché y el grado de asociatividad para una matriz dispersa 10Kx10K con $p_n = 0{,}05$ y $H = 1K$ usando $C_s = 16K$. 55

3.10. Trasposición de una matriz dispersa. 57

3.11. Número de fallos durante la trasposición de una matriz $10K \times 10K$ con $p_n = 0{,}18$ en una caché de 256K palabras en función de L_s para varios grados de asociatividad. 66

3.12. Número de fallos desglosado por vectores durante la trasposición de una matriz $10K \times 10K$ con $p_n = 0{,}18$ en una caché de dos vías y 256K palabras en función de L_s. 67

3.13. Producto matriz dispersa-matriz densa con orden IKJ y *blocking* en los niveles de memoria y registros. 68

3.14. Representación gráfica de las matrices implicadas en el producto matriz dispersa-matriz densa con orden IKJ optimizado y su relación. 69

3.15. Número de fallos medidos y predichos en la caché de datos del primer nivel del procesador R10000 durante el producto de una matriz dispersa 5000x5000 con $p_n = 0{,}004$ y $p_n = 0{,}04$ usando $H = 1000$ para diferentes bloques. 86

3.16. Número de fallos durante el producto de una matriz dispersa 5000x5000 para diferentes niveles de asociatividad en una caché de 128K palabras con un tamaño de línea de 16 palabras; $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$. 88

3.17. Número de fallos durante el producto de una matriz dispersa 5000x5000 para diferentes tamaños de línea en una caché de correspondencia directa de 128K palabras; $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$. 88

3.18. Número de fallos durante el producto de una matriz dispersa 5000x5000 para diferentes tamaños de caché en una caché de dos vías con una línea de 16 palabras; $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$. 89

4.1. Número de fallos durante el producto matriz dispersa-vector de una matriz 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas, $L_{\mathrm{s}} = 8$ y $K = 4$ en función de W y C_{s}. 96

4.2. Número de fallos durante el producto matriz dispersa-vector de una matriz 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas y $W = 8000$ en función del tamaño de la caché y del grado de asociatividad para $L_{\mathrm{s}} = 8$. 96

4.3. Número de fallos durante la trasposición de una matriz dispersa 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas, $L_{\mathrm{s}} = 8$ y $K = 4$ en función de W y C_{s}. 108

4.4. Número de fallos durante la trasposición de una matriz dispersa 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas y $W = 8000$ en función del tamaño de la caché y del grado de asociatividad para $L_{\mathrm{s}} = 8$.109

4.5. Número de fallos durante la trasposición de una matriz dispersa 5K×5K con 125K entradas, $L_{\mathrm{s}} = 4$ y $K = 4$ en función de W y C_{s}. 110

4.6. Número de fallos durante la trasposición de una matriz dispersa 5K×5K con 125K entradas y $W = 200$ en función del tamaño de la caché y del grado de asociatividad para $L_{\mathrm{s}} = 4$. . 111

4.7. Matriz BCSSTM07, perteneciente al conjunto BCSSTRUC1 de la colección Harwell-Boeing. 112

4.8. Matriz CRY10000, perteneciente al conjunto CRYSTAL de la colección NEP, y detalle de la banda. 112

4.9. Probabilidades de que haya un no nulo en las $W + L_s - 1$ filas de la matriz que se extienden sobre $L_s = 2$ columnas consecutivas, siendo $W = 4$. 113

4.10. Un grupo L de L_s columnas consecutivas junto a los $N_D(j)$ grupos a su derecha y los $N_I(i)$ a su izquierda que pueden ser referenciados durante el procesamiento entre las filas i y j de la banda de L. 119

5.1. Código tipo del modelado de códigos densos 130

5.2. Descripción del área regular accedida de una matriz bidimensional `A` durante una iteración del bucle en `K`, en el i-ésimo nivel del anidamiento. 134

5.3. Descripción del área no regular accedida de una matriz bidimensional `A` durante una iteración del bucle en `K`, en el i-ésimo nivel del anidamiento. 134

5.4. Algoritmo de estimación del número de conjuntos compartidos por dos estructuras de datos `A` y `B`. 137

5.5. Algoritmo de estimación del número de líneas que una referencia secuencial aporta a otra. 139

5.6. Construcción típica de bucles con *blocking* 139

5.7. Porciones de una matriz bidimensional `A` con múltiples referencias que han resultado accedidas durante una iteración del bucle en `K`. 143

5.8. Código con múltiples referencias a una matriz 147

5.9. Producto matriz densa-matriz densa con orden JIK. 151

5.10. Código de Stencil. 151

5.11. Método de Jacobi bidimensional. 152

5.12. Producto matriz densa-matriz con *blocking* perfectamente anidado. 152

5.13. Producto matriz densa-matriz con *blocking* y copia con trasposición del bloque. 153

5.14. Código de resolución de ecuaciones diferenciales parciales mediante el método de Gauss-Seidel. 153

Capítulo 1

El análisis de la jerarquía de memoria

El cuello de botella para el rendimiento en los sistemas de computación radica hoy por hoy en la diferencia entre la velocidad del procesador y la del sistema de memoria. En particular, los códigos científicos se encuentran entre los más afectados, pasando algunos de ellos más de la mitad de su tiempo de computación aguardando la recepción de datos de la memoria [37]. Por otra parte, la tendencia actual en el desarrollo de tecnologías de procesamiento y de memoria no lleva sino al aumento de esta diferencia [8], [26], [10]. Así, durante la última década la velocidad de los procesadores ha crecido entre un 50 % y un 100 % cada año, mientras que la de la RAM dinámica crece en torno al 10 % [15],[26].

La técnica comúnmente empleada para reducir las latencias de acceso a memoria, ha sido la construcción de una jerarquía de niveles de memoria en la que éstos, cuanto más cercanos al procesador, son más rápidos pero tienen también una capacidad menor. Los niveles más próximos a la unidad de proceso están constituidos por pequeñas memorias de alto coste con una velocidad similar a la del procesador denominadas cachés, las cuales se basan en los principios de la localidad espacial y temporal [42] para retener el subconjunto de los datos e instrucciones manejados por el programa con mayor probabilidad de acceso en cada momento. La naturaleza secuencial de la mayoría de los accesos a datos y especialmente a las instrucciones fundamenta ambos principios.

En general, la jerarquía de memoria puede constar de varios niveles de caché [4] en los que cada uno de ellos recibe peticiones de acceso a posiciones de memoria del nivel superior (el nivel superior está constituido por los reg-

istros del procesador). Si el nivel contiene la posición solicitada, se dice que la petición resulta en un acierto, transfiriéndose al nivel superior una serie de palabras del tamaño del bloque que éste maneja, en el caso de una lectura, o modificándose el valor contenido por la posición si se trata de una escritura (dependiendo de la política de escritura seguida, el dato también se podría transferir al nivel superior). En caso contrario, la petición resulta en un fallo y el nivel considerado debe solicitar el acceso a la posición al nivel inmediatamente inferior. El tiempo necesario para resolver el fallo se denomina latencia de fallo, y el tiempo durante el que el procesador permanece inactivo debido a los fallos en la jerarquía de memoria, penalización por fallo.

Así pues, resulta crucial para la mejora de las tasas de computación la reducción tanto de la latencia de fallos como de la tasa de fallos, es decir, el porcentaje de referencias que resultan en un fallo, sin olvidar la reducción del tiempo de servicio en los aciertos [26]. Para ello se emplean tanto técnicas hardware en los distintos niveles de la jerarquía (buffers de escritura, asociatividad de la caché) y en el procesador (ejecución en desorden, cambios rápidos de contexto) como software (reestructuración de datos [3] o código [11], etc.), siendo muy activa la investigación en este área. El análisis del comportamiento de la jerarquía de memoria será fundamental para determinar las mejores soluciones técnicas, y para ello disponemos de varios enfoques que repasamos a continuación.

1.1. Estrategias para el estudio del comportamiento de la caché

Podemos distinguir en la bibliografía tres aproximaciones diferentes para el estudio del comportamiento de la caché: simulación software, monitorización hardware y el modelado analítico.

Simulación Software

El enfoque más extendido para conocer el rendimiento de una caché ante un determinado código es la simulación software de la ejecución del mismo, emulando el efecto en la caché de todos y cada uno de los accesos que generaría [47]. Esta técnica, si bien es muy exacta, proporciona poca información sobre las razones del comportamiento de la caché y requiere, en general, tiempos de computación muy grandes, típicamente varias veces mayores que los que requieren los códigos simulados. Se han propuesto varias técnicas para

reducir el tamaño de la secuencia de direcciones (traza) que guía la simulación, a fin de reducir el coste tanto de tiempo de computación como de espacio de disco, el cual puede ser considerable dependiendo de la aplicación. No obstante son pocas las que prestan suficiente atención a la problemática de la reducción de la representatividad que tendrá el subconjunto escogido de la traza.

Monitorización hardware

Algunos de los microprocesadores comercializados en los últimos años superan los problemas asociados a los tiempos de computación y el almacenamiento de trazas proporcionando herramientas de monitorización del rendimiento tales como contadores de fallos incorporados en el chip. Podemos mencionar microprocesadores tales como el MIPS R10000 [52], la familia DEC Alpha [17], Intel Pentium [34], el HP PA-8000 [28] y el IBM Power2 [48]. Sin embargo, su uso presenta las limitaciones de requerir la compilación y ejecución del programa, con lo que sólo pueden proporcionar información a posteriori sobre el sistema, así como su restricción a una plataforma específica. Lo mismo cabe decir de otras formas de medición hardware, como el uso de monitores [14], previas a la aparición de procesadores con estas facilidades.

Modelado analítico

Frente a la imposibilidad de que adolecen estas técnicas para proporcionar rápidamente estimaciones o límites del rendimiento para un amplio abanico de programas y configuraciones de caché, una estrategia más general es la del modelado analítico, el cual, además de requerir tiempos de computación reducidos, flexibiliza el estudio paramétrico del comportamiento de la caché y proporciona más información sobre éste que los métodos experimentales. Su punto débil ha sido tradicionalmente su baja precisión. Muchos modelos extraen parámetros de entrada de trazas [1], [9], [29], [40] y los combinan con los parámetros de definición de la caché, con lo cual su aplicación requiere todavía de la obtención de la trazas, tarea costosa de por sí. No obstante, se han desarrollado modelos más generales que toman como entrada el código a analizar [24], [45], posibilitando de esta forma su uso por parte de compiladores y herramientas de optimización automática.

En la siguiente sección revisaremos algunos de los trabajos relacionados con el estudio de la caché a fin de establecer con mayor claridad el estado actual de la investigación en este campo.

1.2. Trabajos previos

Al igual que para los códigos con patrones de acceso regulares, los accesos irregulares han sido estudiados habitualmente a través de simulaciones guiadas por trazas. Es el caso de trabajos como [43], en donde se simula el producto matriz dispersa-vector usando matrices reales para diferentes configuraciones de caché a fin de concluir cual es la configuración óptima para la ejecución de este algoritmo. El artículo reconoce las limitaciones impuestas por el método escogido para recomendar dicha configuración, es decir, la validez de sus observaciones se limita a las simulaciones efectuadas. De hecho, en nuestro trabajo veremos (apartados 3.2.4 y 4.1.1) que muchas de las características que propone, como el que la caché sea de correspondencia directa, son realmente muy dependientes de los valores que se consideren para la configuración de la caché y los parámetros que definen la matriz utilizada.

Otro trabajo en una línea similar es [53], que propone modificaciones hardware con apoyo software para mejorar el rendimiento caché de aplicaciones con patrones irregulares arbitrarios originados por el uso de registros conectados mediante punteros. El uso de las simulaciones ha dado también lugar a la propuesta de mejoras puramente software para aumentar la tasa de aciertos en la ejecución de algoritmos irregulares [38].

Pasando al campo de los modelos analíticos, [1] y posteriormente [2] obtienen diversos parámetros, tales como las probabilidades de acceso a líneas de código y datos o la probabilidad de que los accesos se den a posiciones de memoria consecutivas, a partir de trazas, orientándose a la estimación del rendimiento en un sistema multiprogramado dando especial atención al sistema operativo. Su enfoque considera tres categorías de fallo: los generados por las referencias iniciales al rellenar la caché, los fallos no estacionarios debidos al cambio de conjunto de trabajo, y los fallos por interferencias debido a las colisiones que considera que se dan aleatoriamente en los conjuntos de la caché.

Otro modelo que también asume que un bloque tiene una probabilidad uniforme de estar asociado a cualquiera de los conjuntos de la caché y que se basa en probabilidades extraídas de trazas concretas es el propuesto en [40], si bien no requiere mediciones específicas para distintos tamaños de línea. Por otra parte, este modelo trata los fallos como un conjunto, y se basa, como haremos nosotros, en el cálculo del área media accedida entre dos referencias a la misma línea. Sin embargo, se orienta a la estimación del número de fallos en el acceso a las instrucciones (y no los datos) de un programa, agrupándolas en bloques.

A diferencia de los dos trabajos anteriores, que consideran cachés de una asociatividad arbitraria, [9] estudia una caché totalmente asociativa con reemplazo LRU. Se basa en el modelo IRM (*Independent Reference Model*) [30], el cual asume que la probabilidad de acceso a cada línea o bloque es constante en el tiempo, lo cual, si bien simplifica el modelo, reduce sustancialmente su ámbito de aplicación, puesto que es una idea opuesta al comportamiento real de los programas, los cuales trabajan en cada momento con un subconjunto de su espacio de instrucciones y de datos que constituye lo que denominamos su conjunto de trabajo.

En [29] se presenta otro modelo analítico basado en trazas, siendo su objetivo la optimización de una jerarquía de memoria completa. En contraste con otros trabajos anteriores de este tipo [33], proporciona una forma general para el tratamiento de la localidad en una carga de trabajo específica, sin embargo, al igual que el modelo precedente asume una asociatividad total, lo cual sigue constituyendo una aproximación poco realista al problema.

En relación a los modelos que no requieren de la existencia de una traza para extraer ninguno de sus parámetros de entrada, un trabajo representativo es [45], que se basa en las ideas presentadas en [44]. Si bien está restringido en su aplicación a cachés de correspondencia directa y a códigos con patrones de acceso regulares, contempla los tres tipos de fallos introducidos en [27]: intrínsecos (accesos por primera vez), de capacidad y de interferencia.

Otro trabajo orientado a códigos densos en cachés de mapeado directo es [24], que deriva ecuaciones para el número total de fallos que se producen en un bucle dado. Este trabajo aprovecha la infraestructura del entorno SUIF [50] como herramienta de análisis automático del código. También proporciona una idea para su extensión a cachés asociativas. El modelo requiere bucles perfectamente anidados en los que las referencias se encuentran en el bucle más interno y utilizan funciones afines en los índices, si bien puede permitir algunos bucles no perfectamente anidados si sólo hay un único bloque básico entre los bucles. Tampoco permite la presencia de sentencias condicionales. A pesar de ello, constata que en los códigos científicos estas son las condicionas más habituales; por ejemplo, aproximadamente el 72 % en los bucles de los códigos del SPECfp las verifican. Además analiza cada bucle por separado, ignorando los efectos que se pueden dar entre ellos. En esta tesis presentaremos una estrategia menos restrictiva para la automatización de nuestro enfoque de modelado.

El único trabajo encontrado referente al modelado analítico basado en el código para aplicaciones con patrones irregulares de acceso a memoria es [46], el cual estudia las auto-interferencias sobre el vector involucrado en el pro-

ducto matriz dispersa-vector en una caché de mapeado directo sin considerar las interferencias con otras estructuras de datos ni intentar derivar un entorno general para modelar este tipo de códigos. Nuestro trabajo contempla todos los tipos de fallos e interferencias y se ha desarrollado haciendo énfasis en la modularidad, es decir, importantes partes del modelo son reusables en diferentes códigos algebraicos, lo que demuestra la potencial aplicabilidad de este tipo de modelado.

1.3. Objetivos y organización de la tesis

El propósito fundamental de esta tesis es el desarrollo de una técnica de modelado analítico basada en probabilidades que permita obtener buenas estimaciones del comportamiento de códigos en una caché arbitraria. Este objetivo tiene una doble vertiente. En primer lugar hemos abordado el desarrollo de un enfoque sistemático y modular para el modelado analítico de códigos con patrones de acceso irregulares. Las bases de este enfoque se asentarán en el capítulo 2. El trabajo derivado del cumplimiento de este primer objetivo nos ha permitido plantear el diseño de una nueva herramienta de optimización automática basada en el modelado analítico de códigos con patrones de acceso regulares.

Nuestro interés por los patrones irregulares viene dado por varios motivos. Por un lado, estos patrones reducen el rendimiento de la jerarquía de memoria, al no respetar los principios de localidad en que ésta se basa. Por otro, tal y como hemos comentado, se han realizado pocos estudios sobre ellos, en parte debido a la mayor complejidad de su análisis debido a su naturaleza, a pesar de que surgen en multitud de aplicaciones. En este trabajo nos centraremos en concreto en aplicaciones numéricas de álgebra matricial dispersa, es decir, que manipulan matrices con un número elevado de posiciones cuyo valor es cero, lo cual lleva a almacenarlas mediante formatos comprimidos que generan este tipo de patrones, como veremos en el capítulo 3 cuando apliquemos nuestro enfoque analítico a códigos concretos. Un aspecto muy importante a tener en cuenta para el modelado de códigos que trabajan con matrices dispersas es la distribución de las posiciones que contienen valores no nulos (entradas). Inicialmente, a lo largo del capítulo 3, consideraremos matrices con una dispersión uniforme de las entradas.

En el capítulo 4 se extenderá el modelo para considerar matrices con distribuciones en banda. A su vez, dentro de este tipo de matrices distinguiremos entre aquellas con una dispersión uniforme de las entradas sobre la

banda, y aquellas que presentan una distribución uniforme por diagonales, es decir, la distribución de entradas en cada una de las diagonales que constituyen la banda puede considerarse uniforme, permitiendo que las diferentes diagonales tengan diferentes densidades de entradas.

Por último, formalizaremos la aplicación de nuestra estrategia de modelado para códigos densos. La regularidad de los patrones de acceso que caracterizan estos códigos permitirá que extendamos nuestra técnica de forma que podamos implementarla en una herramienta de análisis automático de código en el capítulo 5. Esta labor partirá de las bases del modelado expuestas en el capítulo 2 para añadir reglas de obtención del número de fallos por referencia y bucle o las normas para extraer la forma de la región afectada por las referencias a una matriz dada durante la ejecución de un cierto bucle, por ejemplo. El enfoque automatizado se verificará mediante una implementación simplificada que se aplicará a un conjunto de códigos representativos.

Los resultados de este trabajo han sido publicados en [18], [19], [22] y [23] de donde excluímos las conferencias nacionales.

Capítulo 2

Aspectos generales del modelado

En este capítulo introduciremos la terminología y la filosofía en que se basa nuestro enfoque de modelado. También derivaremos fórmulas que se utilizarán asiduamente en los distintos modelos, independientemente de que correspondan a algoritmos con patrones de acceso irregulares, como los que veremos el los capítulos 3 y 4, o a códigos que sólo contienen patrones regulares de referencia a memoria, los cuales estudiaremos en el capítulo 5.

2.1. Parámetros de entrada

Aún cuando la palabra es la unidad física de acceso a memoria, las referencias acceden realmente a valores enteros o en punto flotante y no a palabras. El modelo se construye en función de la unidad de referencia, que se ha elegido como el tamaño de un valor en punto flotante, por ser los más empleados en las estructuras de datos que sufren accesos irregulares en las aplicaciones que hemos estudiado. Así pues, al considerar una caché de C_s palabras con líneas de L_s palabras, realmente queremos decir que la caché puede contener C_s valores en punto flotante. Un parámetro derivado que se usa en algunas fórmulas es $N_\mathrm{c} = C_\mathrm{s}/L_\mathrm{s}$, el número de líneas de la caché.

Los primeros modelos construidos estaban orientados a cachés de correspondencia directa. La posterior extensión a cachés asociativas por conjuntos con política de reemplazo LRU llevó a la inclusión de un nuevo parámetro para describir la caché: el tamaño K del conjunto. A partir de la inclusión de este concepto también es de interés el manejo de $N_\mathrm{k} = N_\mathrm{c}/K$, el número

C_s	Tamaño de la caché en palabras
L_s	Tamaño de la líneas en palabras
N_c	Número de líneas de la caché ($C_\mathrm{s}/L_\mathrm{s}$)
K	Nivel de asociatividad
N_k	Número de conjuntos de la caché (N_c/K)
C_sk	C_s/K
r	$\frac{\text{tamaño de un dato entero}}{\text{tamaño de un dato en punto flotante}}$

Cuadro 2.1: Notación empleada.

de conjuntos de la caché. También se mostrará de utilidad el uso del cociente entre el tamaño de la caché y su nivel de asociatividad $C_\mathrm{sk} = C_\mathrm{s}/K$.

Por último, a fin de tener en cuenta la posibilidad de que haya tipos de datos básicos con un tamaño distinto (enteros, valores en punto flotante de mayor precisión, etc.) pueden utilizarse factores que los escalan. En nuestros modelos hemos considerado un factor r:

$$r = \frac{\text{Tamaño en palabras de un valor entero}}{\text{Tamaño en palabras de un valor en punto flotante}} \tag{2.1}$$

para escalar las referencias a vectores constituidos por enteros.

La tabla 2.1 resume los parámetros aquí introducidos a fin de facilitar su consulta en lo sucesivo.

2.2. Ideas básicas

El modelo considera los tres tipos de fallos existentes en una caché: intrínsecos, auto-interferencias e interferencias cruzadas:

- La primera vez que se accede a un bloque de memoria tiene lugar un fallo intrínseco.
- Las auto-interferencias son fallos sobre líneas que han sido expulsadas previamente de la caché por líneas pertenecientes a la misma estructura de datos.
- Las interferencias cruzadas se refieren a fallos sobre líneas que han sido expulsadas entre dos referencias consecutivas a ellas por líneas pertenecientes a otras estructuras de datos.

Por otra parte, es posible que una línea sea expulsada por una combinación de líneas procedentes de la misma y de otras estructuras de datos, con lo que los dos últimos tipos de fallos pueden unificarse para darles el nombre genérico de interferencias.

La idea en la que se fundamenta nuestro modelo es que si tenemos un conjunto de tamaño K y una política de reemplazo LRU, que es la más frecuente en los sistemas reales, para que una línea sea reemplazada antes de ser reusada es preciso que se acceda a K o más líneas diferentes asociadas al mismo conjunto entre ambos accesos. Obsérvese que cuando $K = 1$ el comportamiento es el de una caché de mapeado directo.

La probabilidad de reemplazo para una línea dada crece con el número de líneas afectadas por los accesos que se dan entre dos referencias consecutivas a ella y depende de la forma en la que estas líneas se distribuyen entre los conjuntos. Esto último viene dado por las posiciones de memoria de las estructuras de datos accedidas, dado que son las que determinan el conjunto de la caché a que está asociada cada línea. Manejamos las áreas cubiertas por los accesos a una estructura de datos dada `V` del programa mediante un vector de área $S_{\mathtt{V}} = S_{\mathtt{V}_0} S_{\mathtt{V}_1} \ldots S_{\mathtt{V}_K}$, donde $S_{\mathtt{V}_0}$ es la proporción en tanto por uno de conjuntos que han recibido K o más líneas de `V`, mientras que $S_{\mathtt{V}_i}, 0 < i \leq K$ es la proporción de conjuntos que han recibido $K - i$ líneas de esta estructura. Esto significa que $S_{\mathtt{V}_i}$ es la proporción de conjuntos de la caché que requieren accesos a otras i nuevas líneas diferentes para reemplazar todas las líneas que contenían cuando se inició el acceso a `V`. En la figura 2.1 tenemos dos ejemplos de vectores de área correspondientes a dos vectores `U` y `V` junto con la distribución de las líneas asociadas a ellos en una caché.

Para calcular el número medio de líneas asociadas al conjunto que nos interesa durante un periodo dado de la ejecución de un código, calcularemos los vectores de área para cada una de las estructuras de datos referenciadas durante ese periodo. Será necesario además proporcionar una operación para sumar los vectores de área y que denominaremos operación de unión. Finalizaremos este apartado con el cálculo del número de líneas de un vector o matriz que compiten con otra por el mismo conjunto de la caché.

2.2.1. Unión de vectores de área

La forma de obtener el vector de área total es una cuestión fundamental. Se calcula como la suma de los vectores de área correspondientes a los accesos a las distintas estructuras de datos del programa. Dados dos vectores de área $S_{\mathtt{U}} = S_{\mathtt{U}_0} S_{\mathtt{U}_1} \ldots S_{\mathtt{U}_K}$ y $S_{\mathtt{V}} = S_{\mathtt{V}_0} S_{\mathtt{V}_1} \ldots S_{\mathtt{V}_K}$ correspondientes a los accesos a las

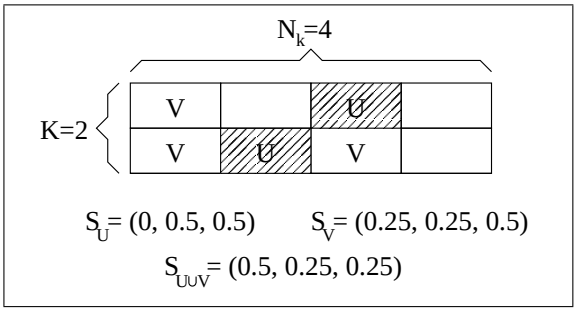

Figura 2.1: Vectores de área correspondientes a accesos un vector `U` y un vector `V` en una caché con $N_{\mathrm{k}} = 4$ y $K = 2$ y vector de área total resultante.

estructuras `U` y `V`, definimos el vector de área unión $S_{\mathtt{U}} \cup S_{\mathtt{V}}$ como:

$$\begin{aligned}(S_{\mathtt{U}} \cup S_{\mathtt{V}})_0 &= \sum_{j=0}^{K} \left(S_{\mathtt{U}_j} \sum_{i=0}^{K-j} S_{\mathtt{V}_i} \right) \\ (S_{\mathtt{U}} \cup S_{\mathtt{V}})_i &= \sum_{j=i}^{K} S_{\mathtt{U}_j} S_{\mathtt{V}_{(K+i-j)}} \qquad 0 < i \leq K\end{aligned} \tag{2.2}$$

donde $(S_{\mathtt{V}} \cup S_{\mathtt{V}})_i, 0 < i \leq K$ es la proporción en tanto por uno de conjuntos que han recibido $K - i$ líneas de estas dos estructuras, y $(S_{\mathtt{V}} \cup S_{\mathtt{V}})_0$ es la proporción de conjuntos que han recibido K o más líneas. La Figura 2.1 ilustra el proceso de unión de vectores de área. En adelante usaremos el símbolo $\cup$ para denotar la operación de unión de vectores de área. Este método no hace ninguna asunción sobre las posiciones relativas de las estructuras de datos en la memoria, puesto que se basa en la suma de las proporciones de área como probabilidades independientes.

2.2.2. Número de líneas en un vector compitiendo por el mismo conjunto de la caché

Se necesita una función para computar el número medio de líneas con las que una línea de una estructura de datos dada compite por el mismo conjunto de la caché para el cálculo de la probabilidad de auto-interferencia. Esta función se define para una estructura con n palabras consecutivas como:

$$\begin{aligned}C(n) &= \lfloor v \rfloor \frac{(v - \lfloor v \rfloor)(\lfloor v \rfloor + 1)}{v} + (\lfloor v \rfloor - 1)\left(1 - \frac{(v - \lfloor v \rfloor)(\lfloor v \rfloor + 1)}{v}\right) \\ &= \frac{\lfloor v \rfloor}{v}(2v - \lfloor v \rfloor - 1)\end{aligned} \tag{2.3}$$

donde $v = n/C_{\text{sk}}$ es el número medio de líneas de la estructura asociadas a un conjunto dado de la caché. Si $v > 1$, en el $(v - \lfloor v \rfloor) \times 100\,\%$ de los conjuntos de la caché el número de líneas del vector o matriz que compiten con otra dada es $\lfloor v \rfloor$. En esta área residen el $\frac{(v-\lfloor v \rfloor)(\lfloor v \rfloor+1)}{v} \times 100\,\%$ de las líneas de la estructura. En el resto de la caché (y por tanto para el resto de la estructura) el número de líneas que compiten es $\lfloor v \rfloor - 1$. A partir de este promedio se obtiene la fórmula (2.3). Además es fácil comprobar que si $v \leq 1$, es decir, cuando la estructura de datos cubre un número de líneas menor que o igual al número de conjuntos de la caché, esta expresión adopta el valor 0, puesto que no hay auto-interferencia posible.

2.3. Patrones de acceso

El cálculo del vector de área depende del patrón de acceso a la estructura de datos a que corresponde, por lo que definiremos una función de vector de área para cada patrón de acceso que encontremos en los códigos analizados. La modularidad del modelo vendrá dada en gran medida por el uso de las expresiones que introduciremos en este apartado para calcular las aportaciones de los accesos a cada estructura de datos a la probabilidad global de generar fallos en función de su patrón de referencia. Los patrones que estudiaremos serán:

- El acceso secuencial, uno de los más frecuentes en todo tipo de códigos.
- El acceso a un vector o matriz en el que todas las líneas tienen una probabilidad uniforme de ser referenciadas. Es el acceso irregular más sencillo que encontraremos.
- El acceso a grupos de elementos con una probabilidad uniforme de referencia. La diferencia con el anterior radicará en dos aspectos: el primero es que los grupos de elementos en los que se concentrará la probabilidad uniforme de referencia no se corresponderán a líneas de caché sino que tendrán un tamaño arbitrario aunque igual para todos ellos. El segundo es que en cada acceso a cada grupo sólo se accederá a un dato y en accesos sucesivos a un grupo dado se accederá a palabras consecutivas.
- Un patrón similar al anterior que restringe la probabilidad de referencia en cada acceso a una serie de grupos consecutivos del vector o matriz. Este patrón se construye de forma que en cada nuevo acceso hay un

```
DO J=1, M
  DO I=1, N
    C(J)=C(J)+A(I,J)
  ENDDO
ENDDO
```

Figura 2.2: Código constituido únicamente por accesos secuenciales.

grupo que deja de pertenecer al conjunto de grupos accesibles y hay uno nuevo que se incorpora.

- El acceso a regiones de palabras consecutivas separadas entre sí por una distancia constante (*stride*) es un patrón regular que aparece tanto en códigos de álgebra densa como en algunos de los que estudiaremos en el siguiente capítulo.

2.3.1. Acceso secuencial

El patrón más común y más sencillo de modelar es el secuencial, que viene ilustrado por el código FORTRAN de la figura 2.2. Cabe recordar que en este lenguaje las matrices bidimensionales, como `A` en la figura, se almacenan por columnas. El vector de área de interferencia cruzada para un acceso a n posiciones consecutivas de memoria es $S_{\mathrm{s}}(n)$:

$$\begin{array}{l} S_{\mathrm{s}_{(K-\lfloor l \rfloor)}}(n) = 1 - (l - \lfloor l \rfloor) \\ S_{\mathrm{s}_{(K-\lfloor l \rfloor - 1)}}(n) = l - \lfloor l \rfloor \\ S_{\mathrm{s}_i}(n) = 0 \quad 0 \leq i < K - \lfloor l \rfloor - 1, K - \lfloor l \rfloor < i \leq K \end{array} \tag{2.4}$$

donde $l = (n + L_{\mathrm{s}} - 1)/C_{\mathrm{sk}}$ es el número medio de líneas que corresponden a cada conjunto. Si $l \geq K$ entonces $S_{\mathrm{s}_0} = 1, S_{\mathrm{s}_i} = 0, 0 < i \leq K$, dado que todos los conjuntos reciben una media de K o más líneas. El término $L_{\mathrm{s}} - 1$ sumado a n representa el número medio de palabras extra que se traen a la caché en la primera y última de las líneas accedidas. La figura 2.3 muestra la distribución en una caché de las líneas de un vector accedido secuencialmente y el vector de área asociado S_{s}.

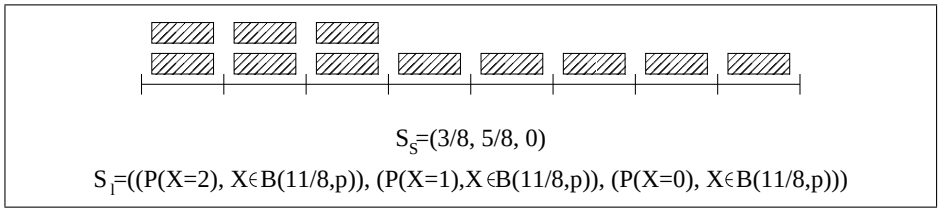

Figura 2.3: Vectores de área correspondientes a un acceso secuencial y a un acceso a líneas con una probabilidad uniforme de referencia p de un vector que cubre 11 líneas de una caché con $N_{\mathrm{k}} = 8$ y $K = 2$.

```
DO J=1, M
  A(J)=ABS(X(C(J)))
ENDDO
```

Figura 2.4: Código con un acceso con una probabilidad uniforme de referencia por línea para el vector `X`.

Vector de área de auto-interferencia

El vector de área de auto-interferencia de este acceso viene dado por la expresión $S_{\mathrm{s}}(C(n)C_{\mathrm{sk}})$, ya que la auto-interferencia experimentada por cada línea equivale a la interferencia cruzada producida por el acceso a un vector de $C(n)C_{\mathrm{sk}}$ palabras. Este vector aportaría $C(n)$ líneas a cada conjunto, que es precisamente el número medio de líneas del vector estudiado con que cada línea compite en el conjunto al que está asociada.

2.3.2. Acceso a líneas con una probabilidad uniforme de referencia

Para ilustrar este patrón irregular, que es el más común en los códigos que hemos analizado, utilizaremos el código de la figura 2.4. Supongamos que tanto `A` como `C` son dos vectores de tamaño `M`, constando éste último de enteros con una distribución uniforme entre 1 y `N`, el tamaño del vector `X`. No sabremos exactamente qué líneas de `X` se acceden si no ejecutamos o simulamos este bucle debido a la referencia indirecta a través de los valores de `C`. Sin embargo, partiendo de que los valores de este vector tienen una distribución uniforme a lo largo de la longitud de `X`, sí podemos estimar cuántas líneas se habrán accedido por término medio y cómo pueden estar

distribuidas en la caché. El vector de área para un acceso a un vector de n palabras en donde cada una de las líneas de caché en las que se divide tiene la misma probabilidad P_l de ser accedida, $S_\mathrm{l}(n, P_\mathrm{l})$, puede calcularse como:

$$\begin{array}{ll} S_{\mathrm{l}_i}(n, P_\mathrm{l}) = P(X = K - i) & m < i \leq K \\ S_{\mathrm{l}_m}(n, P_\mathrm{l}) = P(X \geq K - m) & \\ S_{\mathrm{l}_i}(n, P_\mathrm{l}) = 0 & 0 \leq i < m \end{array} \tag{2.5}$$

donde $X \in B(n/C_\mathrm{sk}, P_\mathrm{l})$, siendo B($n$,$p$) la distribución binomial[1] y $m = \mathrm{máx}\{0, K - \lceil n/C_\mathrm{sk} \rceil\}$. Esta fórmula se basa en que si el vector consta de n palabras, habrá por término medio n/C_sk líneas del vector asociadas a cada conjunto de la caché. Dado que el vector es una región de memoria consecutiva, el máximo de líneas que podría aportar a un conjunto durante su acceso sería $\lceil n/C_\mathrm{sk} \rceil$, lo cual hace que las posiciones del vector $S_{\mathrm{l}_i}(n, P_\mathrm{l})$ para $0 \leq i < m$ sean forzosamente cero. A partir de ahí, debido a la distribución uniforme de probabilidad sabemos que el número de líneas por conjunto pertenece a una binomial $B(n/C_\mathrm{sk}, P_\mathrm{l})$, y dado que la posición i (para $i > 0$) del vector representa la porción de conjuntos que han recibido $K - i$ líneas, su valor será la probabilidad de que el valor de la variable asociada a dicha binomial sea $K - i$. La posición más baja del vector con probabilidad de no ser nula (m) agrupará la probabilidad de que el número de líneas accedidas sea $K - m$ o mayor.

Se muestra un ejemplo de este tipo de acceso y del acceso secuencial en la Figura 2.3 usando un vector que ocupa 11 líneas con una probabilidad uniforme de referencia genérica p en una caché con 8 conjuntos y un nivel de asociatividad 2. En el caso de k accesos de este tipo el vector de área es $S^k_\mathrm{l}(n, P_\mathrm{l}) = S_\mathrm{l}(n, 1 - (1 - P_\mathrm{l})^k)$.

Vector de área de auto-interferencia

El vector de área de auto-interferencia para este acceso viene dado por:

$$S^k_\mathrm{la}(n, P_\mathrm{l}) = S^k_\mathrm{l}(C(n)C_\mathrm{sk}, P_\mathrm{l}) \tag{2.6}$$

siendo el razonamiento similar al expuesto en el apartado anterior.

[1] Definimos la distribución binomial sobre un número no entero de elementos n como $P(X = x), X \in B(n, p) = (P(X = x), X \in B(\lfloor n \rfloor, p))(1 - (n - \lfloor n \rfloor)) + (P(X = x), X \in B(\lceil n \rceil, p))(n - \lfloor n \rfloor)$

```
DO K=1, M
  J=C(K)
  P=R(J)
  A(K)=2*X(P)+1
  R(J)=P+1
ENDDO
```

Figura 2.5: Código con un acceso a grupos de elementos con una probabilidad uniforme de referencia para el vector `X`.

2.3.3. Acceso a grupos de elementos con una probabilidad uniforme de referencia

En la trasposición de una matriz dispersa encontramos el caso de un vector de n palabras dividido en grupos de t palabras donde la probabilidad P_{g} de acceder a cada grupo es la misma. Esto ocurre en el bucle principal del algoritmo, que accede a las entradas de la matriz de entrada por filas (secuencialmente) trasladándolas al grupo correspondiente a la columna a la que pertenecen en los vectores que definen la matriz resultado. Cada grupo tiene tantas posiciones como entradas tiene la columna asociada en la matriz de entrada. Durante el procesamiento de cada fila de la matriz de entrada sólo se accede a una de las líneas del grupo, y accesos consecutivos al mismo grupo referencian posiciones de memoria consecutivas. El código de la figura 2.5 tiene un acceso de este tipo sobre `X` si suponemos que `C` consta de enteros repartidos uniformemente entre 1 y el número de grupos de que consta el vector `X`, sin orden alguno, y que la posición i-ésima del vector `R` indica el punto de comienzo del i-ésimo grupo al iniciarse el bucle. El área cubierta por un acceso de este tipo se describe mediante el vector de área $S_{\mathrm{g}}(n, t, P_{\mathrm{g}})$:

$$S_{\mathrm{g}}(n,t,P_{\mathrm{g}}) = \begin{cases} S_{\mathrm{l}}(n, 1-(1-P_g)^{L_{\mathrm{s}}/t}) & \text{si } t \leq L_{\mathrm{s}} \\ S_{\mathrm{l}}(n, P_{\mathrm{g}}L_{\mathrm{s}}/t) & \text{si } t > L_{\mathrm{s}} \end{cases} \tag{2.7}$$

donde si $t \leq L_{\mathrm{s}}$, cada línea del vector tiene una probabilidad uniforme $1-(1-P_g)^{L_{\mathrm{s}}/t}$ de ser accedida. Por el contrario, si $t > L_{\mathrm{s}}$ esta probabilidad es $P_{\mathrm{g}}L_{\mathrm{s}}/t$.

Como en el caso de S_{l}, S_{g} puede extenderse para calcular el área cubierta

por k accesos[2]:

$$S_{\mathrm{g}}^{k}(n,t,P_{\mathrm{g}}) = \begin{cases} S_{\mathrm{l}}(n, 1-(1-P_g)^{kL_{\mathrm{s}}/t}) & \text{si } t \leq L_{\mathrm{s}} \\ S_{\mathrm{l}}(n, L_{\mathrm{g}}^{k}(t,P_g)\frac{L_{\mathrm{s}}}{t}) & \text{si } t > L_{\mathrm{s}} \end{cases} \tag{2.8}$$

donde $L_{\mathrm{g}}^{k}(t,P_{\mathrm{g}})$ es el número medio de líneas accedidas de cada grupo tras k accesos:

$$L_{\mathrm{g}}^{k}(t,P_{\mathrm{g}}) = \text{mín}\left\{ t/L_{\mathrm{s}}, \sum_{i=1}^{k} \left(\begin{array}{c} k \\ i \end{array} \right) P_g^i (1-P_g)^{(k-i)} \left(1 + \frac{i-1}{L_{\mathrm{s}}}\right) \right\} \tag{2.9}$$

ya que en este caso L_{s} accesos consecutivos referencian la misma línea, y sólo L_{s} accesos después del primero a cada línea del grupo se accede a una nueva. La fórmula consiste en sumar las probabilidades de que el número de accesos a un grupo dado sea $1, 2, \ldots, k$ multiplicando cada una por el número medio de líneas que ese número de accesos supone: si hay un acceso se ha accedido al menos a una línea; por cada acceso adicional se accede por término medio a $1/L_{\mathrm{s}}$ de una nueva. La probabilidad de que tras k accesos globales al vector i hayan recaído en un grupo dado es $(P(X=i), X \in B(k,P_{\mathrm{g}})))$, dado que en cada acceso hay una probabilidad uniforme P_{g} de acceder a cada grupo. Esta expresión puede llegar a ser mayor que t/L_{s}, el número de líneas de que consta un grupo, debido a que puede contabilizar la traída a la caché de líneas en las que hay elementos del grupo pero que están compartidas con otros. De ahí la necesidad de tomar el menor de los dos valores como número real medio de líneas accedidas para un solo grupo. Finalmente, $L_{\mathrm{g}}^{k}(t,P_{\mathrm{g}})$ debe multiplicarse por L_{s}/t en (2.8) para convertirlo en una probabilidad de acceso por línea.

Vector de área de auto-interferencia

El vector de área de auto-interferencia correspondiente a un acceso de este tipo puede estimarse como:

$$S_{\mathrm{ga}}^{k}(n,t,P_{\mathrm{g}}) = S_{\mathrm{l}}(C(n)C_{\mathrm{sk}}, 1-(S_{\mathrm{g}_K}^{k}(n,t,P_{\mathrm{g}}))^{C_{\mathrm{sk}}/n}) \tag{2.10}$$

puesto que, como hemos visto en los accesos anteriores, dada un área de n palabras, cada línea de la misma compite con $C(n)$ líneas de media en

[2]En la expresión (2.8) k es un entero. En caso contrario, $S_{\mathrm{g}}^{k}(n,t,p)$ se calcula mediante la interpolación:

$$S_{\mathrm{g}}^{k}(n,t,p) \approx S_{\mathrm{g}}^{\lfloor k \rfloor}(n,t,p) + (k - \lfloor k \rfloor)(S_{\mathrm{g}}^{\lfloor k \rfloor+1}(n,t,p) - S_{\mathrm{g}}^{\lfloor k \rfloor}(n,t,p))$$

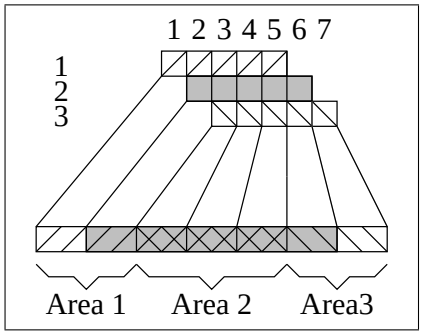

Figura 2.6: Correspondencia entre la localización de las entradas en una matriz banda a trasponer y los grupos a donde son copiadas en los vectores que definen la matriz traspuesta.

su conjunto, lo cual equivale a un acceso a un vector de $C(n)C_{\mathrm{sk}}$ palabras. En cuanto a la probabilidad de acceso por línea, recuérdese que el elemento K de un vector de área contiene la porción de conjuntos de la caché que no han recibido ninguna línea tras la aplicación de un determinado tipo de acceso. Por tanto, $S^k_{\mathrm{g}_K}(n,t,P_{\mathrm{g}})$ es la probabilidad por conjunto de que no se haya recibido ninguna línea. Como $S^k_{\mathrm{g}}(n,t,P_{\mathrm{g}})$ se construye en función de $S_{\mathrm{l}}(n,F(t,P_{\mathrm{g}}))$, donde F es una función que adopta los valores indicados en la expresión (2.8), el componente K se obtendrá como $(1-F(P_{\mathrm{g}}))^{n/C_{\mathrm{sk}}}$. De ahí que al elevar este valor a C_{sk}/n obtengamos la probabilidad de que una línea no haya sido accedida, y sustrayendo este valor de 1, tengamos la probabilidad de acceso por línea $F(t,P_{\mathrm{g}})$.

2.3.4. Acceso a áreas desplazadas con referencias sucesivas

El modelo puede extenderse para considerar el caso en el que sólo un subconjunto de grupos adyacentes del vector sea referenciable en cada acceso, desplazándose este subconjunto con las sucesivas referencias. Esto ocurre en la trasposición de una matriz banda dispersa: durante el procesamiento de cada fila la probabilidad de acceso sólo se distribuye entre aquellos grupos cuyas columnas pertenecen a la banda, y en el procesamiento de cada nueva fila, una columna abandona la banda por la izquierda, perdiendo por tanto su grupo toda probabilidad de acceso, y otra se incorpora por la derecha, añadiendo su grupo de elementos al conjunto de elegibles. Esta situación se muestra en la figura 2.6 con una banda de 5 columnas: durante el procesamiento de la primera fila pueden resultar accedidos los grupos 1 a 5 si hay

una entrada en columna asociada, mientras que en la segunda fila estos grupos son los del 2 al 6, y en la tercera sólo puede accederse a los grupos del 3 al 7.

Sea $S_{gb}(b, t, P_{\mathrm{g}})$ el vector de área correspondiente a un acceso a b grupos consecutivos de t elementos con una probabilidad uniforme por grupo P_{g} de referenciar uno y sólo uno de los elementos del grupo. Este valor viene dado por $S_{\mathrm{g}}(bt, t, P_{\mathrm{g}})$, como podemos deducir de lo visto en el apartado anterior. Definimos $S^k_{gb}(b, t, P_{\mathrm{g}})$ como el vector de área correspondiente a k accesos de este tipo en los que se da un desplazamiento como el arriba descrito del conjunto de grupos referenciables con cada nuevo acceso. Los accesos que siguen este patrón cubren tres áreas adyacentes:

1. Un área formada por $M_{kb} = \mathrm{mín}\{k, b\} - 1$ grupos, que corresponden a $L_v = M_{kb}/G_{\mathrm{l}}$ líneas, siendo $G_{\mathrm{l}} = L_{\mathrm{s}}/t$ el número medio de grupos por línea. Estas líneas tienen una probabilidad creciente de ser accedidas (Área 1 en la Figura 2.6). La probabilidad media de acceso de estas líneas puede calcularse como:

$$P^k_v(b, t, P_{\mathrm{g}}) = \begin{cases} 1 - (1 - P_{\mathrm{g}})^{G_{\mathrm{l}}(G_{\mathrm{l}}+1)/2} \frac{1-P_t^{L_v}}{(1-P_t)L_v} & \text{si } t < L_{\mathrm{s}} \\ \frac{1}{L_v} \sum_{i=1}^{M_{kb}} (1 - (1 - P_{\mathrm{g}})^i + \sum_{j=0}^{\lfloor (i-L_{\mathrm{s}}/2)/L_{\mathrm{s}} \rfloor} P_{>}(i, P_{\mathrm{g}}, j)) & \text{si } t \geq L_{\mathrm{s}} \end{cases} \tag{2.11}$$

donde $P_t = (1 - P_{\mathrm{g}})^{G_{\mathrm{l}}^2}$ y $P_{>}(i, p, j) = P(X > L_{\mathrm{s}}/2 + jL_{\mathrm{s}}), X \in B(i, p)$.

El sentido de esta ecuación es el siguiente: supongamos en primer lugar que $t < L_{\mathrm{s}}$. Si consideramos los últimos G_{l} accesos donde hay una probabilidad de acceder una línea dada, en el último sólo podría haberse accedido a uno de los grupos, en el penúltimo a dos, etc. Por tanto en esos últimos G_{l} accesos habría en total[3] $1 + 2 + \ldots + G_{\mathrm{l}} = G_{\mathrm{l}}(G_{\mathrm{l}} + 1)/2$ posibles accesos a los grupos que contiene, con lo que la probabilidad de no haber accedido a la línea durante ellos sería $(1 - P_{\mathrm{g}})^{G_{\mathrm{l}}(G_{\mathrm{l}}+1)/2}$. A partir del acceso G_{l}-ésimo anterior al actual hacia atrás, todos los grupos de la línea tienen probabilidad P_{g} de ser accedidos. Dado que con cada acceso nos desplazamos también un grupo hacia delante, para cada nueva línea tendríamos G_{l}^2 nuevos posibles accesos a grupos. Así, si numerásemos las líneas de esta zona de 0 a $L_v - 1$ tendríamos finalmente que la probabilidad de que la línea i-ésima hubiese recibido algún acceso sería:

$$P_{\mathrm{acc}\ t<L_{\mathrm{s}}}(i) = 1 - (1 - P_{\mathrm{g}})^{G_{\mathrm{l}}(G_{\mathrm{l}}+1)/2 + iG_{\mathrm{l}}^2} \tag{2.12}$$

[3] Recuérdese que dado el entero i, $1 + 2 + \ldots + i = i(i + 1)/2$. En este caso G_{l} no tiene por qué ser un entero, pero el valor obtenido ha demostrado ser una buena aproximación

La expresión del primer caso en 2.11 es la media aritmética de estos valores. Esta estrategia simplifica el problema al considerar los $M_k b$ accesos en grupos de G_l, como puede observarse, cuando realmente no tiene por qué ser un múltiplo de ese valor. No obstante, el error introducido es pequeño, pues aunque puede ser grande porcentualmente para valores pequeños de $M_k b$, en estos casos esta área consta de pocas líneas.

El método de cálculo para el caso de que $t \geq L_\mathrm{s}$ es el inverso: se calcula para cada grupo el número medio de líneas de que consta que han sido accedidas y se divide entre L_v para tener la probabilidad media de acceso por línea. El término i-ésimo del sumatorio da el número medio de líneas accedidas en el i-ésimo grupo, el cual experimenta i iteraciones en las que puede ser accedido. Por tanto, se accede al menos a una línea con una probabilidad $1 - (1 - P_\mathrm{g})^i$.

Por otra parte, el número de accesos al grupo pertenece a una binomial $B(i, P_\mathrm{g})$. Situando la posición inicial por término medio del primer punto accesible de cada grupo en el intervalo estudiado en la posición media de una línea de caché, si el número de accesos es mayor que $L_\mathrm{s}/2$ se accedería a una segunda línea. Siguiendo la misma lógica, si el número de accesos fuese mayor que $L_\mathrm{s} + L_\mathrm{s}/2$ se accedería a una tercera, y así sucesivamente. Este es el cálculo de probabilidades que se efectúa en el segundo sumatorio, que tiene una iteración para cada una de las posibles líneas sobre las que se puede extender el área accesible en el intervalo estudiado descontado las primeras $L_\mathrm{s}/2$ posiciones que corresponderán por término medio a la primera línea.

2. Un conjunto de $L_f = |k - b + 1|/G_\mathrm{l}$ líneas con una probabilidad constante de acceso $P_f^k(b, t, P_\mathrm{g})$ (Área 2 en la Figura 2.6):

$$P_f^k(b,t,P_\mathrm{g}) = \begin{cases} 1 - (1 - P_\mathrm{g})^{M_{kb} G_\mathrm{l}} & \text{si } t < L_\mathrm{s} \\ G_\mathrm{l}(1 - (1 - P_\mathrm{g})^{M_{kb}} + \displaystyle\sum_{j=0}^{\lfloor (M_{kb} - L_\mathrm{s}/2)/L_\mathrm{s} \rfloor} P_>(M_{kb}, P_\mathrm{g}, j)) & \text{si } t \geq L_\mathrm{s} \end{cases} \tag{2.13}$$

La zona de probabilidad constante está constituida por grupos sobre los que se puede haber accedido M_{kb} veces, puesto que es imposible tener más de los k accesos considerados, y por otra parte, cada grupo sólo puede ser accesible como mucho durante b accesos. Así, cuando $t < L_\mathrm{s}$, en los M_{kb} accesos los G_l grupos de cada línea son accesibles, con lo que la probabilidad de acceso es la expuesta en (2.13).

En el caso de $t \geq L_s$ utilizamos la expresión ya introducida en (2.11) que calcula el número medio de líneas accedidas por grupo, para $i = M_{kb}$ accesos. Dicho valor se multiplica por G_l para escalarla a probabilidad.

Para $k \geq B$, en el caso de la trasposición de una matriz dispersa en banda, que es el único en el que hemos encontrado este patrón, $P_f^k(b, t, P_g)$ debe ser 1. El motivo es que al finalizar el tratamiento del área de la banda a la que pertenece un grupo dado, forzosamente éste ha sido accedido en su totalidad. La pequeña desviación con respecto a lo valores que se obtendrían aplicando (2.13) se debe a que para simplificar el tratamiento matemático, hemos considerado que el número de accesos reales a un grupo tras k iteraciones pertenece a una binomial $B(i, P_g)$, como ya hemos comentado. Sin embargo, puede observarse que realmente sigue una distribución hipergeométrica, cuya utilización implica el cálculo de probabilidades empleando números combinatorios, computacionalmente costosos.

3. Un último conjunto simétrico con respecto al primero y del mismo tamaño en donde las probabilidades de acceso son idénticas pero decrecientes (Área 3 en la Figura 2.6).

Una vez que se han calculado las probabilidades de acceso medias para las líneas de cada una de estas tres regiones consecutivas, sólo resta combinarlas para obtener el correspondiente vector de área.

Vector de área de auto-interferencia

El vector de área de auto-interferencia puede estimarse mediante la expresión:

$$S^k_{\mathrm{gba}}(b, t, P_g) = S_l(C(t(b+k-1))C_{sk}, 1 - (S^k_{\mathrm{gb}_K}(b, t, P_g))^{C_{sk}/(t(b+k-1))}) \quad (2.14)$$

El razonamiento es análogo al expuesto al explicar el origen de la expresión del cálculo del vector de área de auto-interferencia en el apartado anterior. La diferencia radica en que en este caso, tras k accesos no se ha accedido a n palabras sino a $t(L_f + 2L_v)G_l = t(b + k - 1)$.

2.3.5. Acceso a regiones separadas por una distancia constante

Un tipo de acceso regular frecuente es el consistente en el acceso a N_R regiones constituidas cada una por T_R palabras consecutivas, habiendo una

```
R=0
DO I=1, M
  DO J=1, N
    R=R+A(I,J)
  ENDDO
ENDDO
```

Figura 2.7: Código con accesos a regiones separadas por una distancia constante.

distancia (*stride*) entre cada par de regiones de L_{R} palabras. Por ejemplo, en la figura 2.7 cada vez que se completa una iteración del bucle en `I`, se ha accedido a `N` regiones de la matriz `A` que constan de un elemento y cuyas posiciones de inicio están separadas entre sí por `M` datos; el tamaño de cada columna. Aunque se han propuesto aproximaciones analíticas para el cálculo del área afectada por un acceso de este tipo en cachés de correspondencia directa [45], éstas simplifican el problema y no conocemos generalizaciones para cachés asociativas por conjuntos. Por ello hemos preferido usar en este caso una aproximación mixta que comienza realizando una simulación simplificada del acceso a partir de la cual se extraen las medias analíticas del número de líneas que recaerían en cada conjunto de la caché. El vector de área correspondiente se obtiene posteriormente a partir de estos valores.

En un primer paso, se calculan las posiciones C_i y F_i que corresponderán al comienzo y al final, respectivamente, de las regiones en una caché de tamaño C_{sk}, considerando que:

$$\begin{aligned} &C_0 = 0 \\ &C_i = (C_{i-1} + L_{\mathrm{R}}) \bmod (C_{\mathrm{s}}/K), 0 < i \leq N_{\mathrm{R}} \\ &F_i = (C_i + T_{\mathrm{R}} - 1) \bmod (C_{\mathrm{s}}/K), 0 \leq i \leq N_{\mathrm{R}} \end{aligned} \tag{2.15}$$

En dos vectores CV y FV de tamaño C_{sk} inicializados a cero se suma una unidad en cada posición a la que corresponda un C_i o un F_i, respectivamente. A continuación se los recorre calculando la media de líneas del acceso que corresponden a cada conjunto de L_{s} posiciones de estos vectores, esto es, a una línea de un conjunto de la caché. Para ello se utilizan tres valores. El primero viene dado por:

$$L_{\mathrm{G}}(0) = \lfloor (T_{\mathrm{R}} - 1)/C_{\mathrm{sk}} \rfloor N_{\mathrm{R}} + \sum_{i=C_{\mathrm{sk}}-(T_{\mathrm{R}}-1) \bmod C_{\mathrm{sk}}}^{C_{\mathrm{sk}}-1} \mathrm{CV}(i) \tag{2.16}$$

que indica el número de líneas correspondientes a diversas regiones que está garantizado que van a estar asociadas al primer conjunto de la caché considerada. Estas líneas proceden de todas las regiones si $T_{\mathrm{R}} \geq C_{\mathrm{sk}}$, que es lo que aporta el primer componente de la suma; y/o simplemente de las regiones que comienzan en los conjuntos precedentes y cuyo fin no se ha alcanzado. Para un conjunto que se iniciase en la posición j este valor se actualiza como:

$$L_{\mathrm{G}}(j) = L_{\mathrm{G}}(j-1) + \mathrm{C}V(j-1) - \mathrm{F}V(j-1) \tag{2.17}$$

Por otro lado tendremos $L_{\mathrm{F}}(j)$, el número medio de líneas correspondientes a regiones que finalizan en el conjunto que comienza en la posición j de la caché habiendo tomando $C_0 = 0$, pero que con desplazamientos $C_0 = 1, \ldots L_{\mathrm{s}} - 1$ podrían finalizar en el conjunto siguiente. Se calcula como:

$$L_{\mathrm{F}}(j) = \sum_{i=j}^{j+L_{\mathrm{s}}-1} \mathrm{F}V(i)(i-j)/L_{\mathrm{s}} \tag{2.18}$$

Para calcular el número total de líneas asociadas a un conjunto, hay que contabilizar las regiones que se inician en él. Ello requiere utilizar un peso similar al usado en la fórmula (2.18) para tener en cuenta la posibilidad de que con posiciones de inicio diferentes para C_0 la región comenzase en el conjunto siguiente. Este valor L_{C} se calcularía para un conjunto que se inicie en la posición j como:

$$L_{\mathrm{C}}(j) = \sum_{i=j}^{j+L_{\mathrm{s}}-1} \mathrm{C}V(i)(L_{\mathrm{s}} - (i-j))/L_{\mathrm{s}} \tag{2.19}$$

Disponiendo de estas ecuaciones puede calcularse la media de líneas asociadas al conjunto de la caché que comienza en la posición $j = 0, L_{\mathrm{s}}, \ldots, C_{\mathrm{sk}} - L_{\mathrm{s}}$ como:

$$L(j) = L_{\mathrm{G}}(j) + L_{\mathrm{F}}((j + C_{\mathrm{sk}} - L_{\mathrm{s}}) \bmod C_{\mathrm{sk}}) + L_{\mathrm{C}}(j) \tag{2.20}$$

Finalmente el vector de área de la interferencia cruzada correspondiente a este acceso se calcula a partir de estos valores como:

$$S_{\mathrm{r}}(N_{\mathrm{R}}, T_{\mathrm{R}}, L_{\mathrm{R}}) = \frac{1}{N_{\mathrm{k}}} \sum_{i=0}^{N_{\mathrm{k}}-1} S_{\mathrm{s}}(L(iL_{\mathrm{s}})C_{\mathrm{sk}}) \tag{2.21}$$

puesto que en el i-ésimo conjunto recaen por término medio $L(iL_{\mathrm{s}})$ líneas, y el vector de área asociado a una interferencia con n líneas ya hemos comentado en otras ocasiones que corresponde a $S_{\mathrm{s}}(nC_{\mathrm{sk}})$.

Vector de área de auto-interferencia

Si se trata del cálculo del vector de área de auto-interferencia se opera de la siguiente forma:

$$S_{\mathrm{ra}}(N_{\mathrm{R}}, T_{\mathrm{R}}, L_{\mathrm{R}}) = \frac{\sum_{i=0}^{N_{\mathrm{k}}-1} S_{\mathrm{s}}(\mathrm{máx}\{0, L(iL_{\mathrm{s}}) - 1\}C_{\mathrm{sk}})L(iL_{\mathrm{s}})}{\sum_{i=0}^{N_{\mathrm{k}}-1} L(iL_{\mathrm{s}})} \tag{2.22}$$

Así pues, se aplica la misma idea teniendo en cuenta que cada línea en el conjunto i-ésimo compite con $L(iL_{\mathrm{s}}) - 1$. El vector de área de auto-interferencia para cada conjunto se multiplica por el número de líneas del acceso que recaen en él para obtener al final el vector promediado por línea.

Vector de área de auto-interferencia en accesos sucesivos con diferente posición de inicio de las regiones

Hay un caso particular de este tipo de acceso que merece mención aparte. Se trata del cálculo del vector de área de auto-interferencia cuando en una iteración la dirección de inicio de cada región no es la de la iteración anterior, sino que comienza una posición de memoria después. Este acceso se daría por ejemplo en el código FORTRAN de la figura 2.7, donde en cada iteración del bucle en `I` se accede a una fila de la matriz `A`. Si queremos calcular el vector de área de auto-interferencia que genera este acceso para calcular la probabilidad de fallo al acceder a `A(I+1,J)` tras haber accedido en la iteración anterior a `A(I,J)`, residiendo ambos en la misma línea, habremos de tener en cuenta que entre ambos accesos se han leído las posiciones `A(I,J+1)`, `A(I,J+2)`, ..., `A(I,N)` y `A(I+1,1)`, `A(I+2,1)`, ..., `A(I+1,J-1)`.

El vector de auto-interferencia que se generaría empleando la técnica descrita para el acceso a regiones separadas por una distancia constante sería sin embargo la correspondiente a los accesos a todos los elementos de una misma fila. Como puede observarse es un caso habitual, y se ha comprobado que la simplificación del problema consistente en obviar la distancia entre las posiciones de inicio de las regiones en las dos iteraciones y aplicar el procedimiento descrito en el apartado anterior no proporciona buenas estimaciones.

Además de los vectores `FV`, `CV` y de los valores $L_{\mathrm{G}}(j)$ ya introducidos, utilizaremos un vector `T` de C_{sk} elementos en el que todas las posiciones

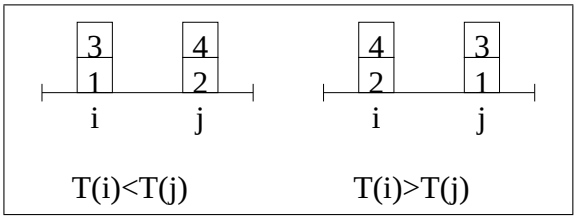

Figura 2.8: Orden de acceso de las regiones que finalizan en las posiciones i y j en función de la relación entre $\mathtt{T}(i)$ y $\mathtt{T}(j)$.

valen cero excepto aquellas para las que $\mathtt{FV}(j) \neq 0$, las cuales adoptarán el valor $\mathtt{T}(j) = \text{mín}\{i/F_i = j\}$. Así pues, T permite comparar grupos de regiones para determinar su orden de acceso como muestra la figura 2.8. En cada punto en donde $\mathtt{FV}(j) \neq 0$ se efectuará el siguiente cálculo para los valores $h = 0, \ldots, L_\mathrm{s} - 2$:

$$\begin{aligned}\bar{L}(j,h) =& L_\mathrm{G}(j) - 1 + \sum_{i=j}^{j+L_\mathrm{s}-2-h} \mathrm{C}V(i) + \sum_{i=j-h+1}^{j} \mathrm{F}V(i) + \\ & \bar{C}(j, j+L_\mathrm{s}-1-h) + \bar{F}(j, j-h-1)\end{aligned} \tag{2.23}$$

en donde las funciones $\bar{C}(j,i)$ y $\bar{F}(j,i)$ se calculan mediante las expresiones:

$$\begin{aligned}\bar{C}(j,i) =& \text{mín}\{1, \mathrm{C}V(i)\}(\mathrm{C}V(i) - \mathrm{Menor}(\mathrm{T}(i+T_\mathrm{R}-1), \mathrm{T}(j)) + \\ & (1 - \mathrm{F}V(j))/2)\end{aligned} \tag{2.24}$$

$$\bar{F}(j,i) = \text{mín}\{1, \mathrm{F}V(i)\}(\mathrm{Menor}(\mathrm{T}(i), \mathrm{T}(j)) + (\mathrm{F}V(j) - 1)/2) \tag{2.25}$$

en donde para simplificar las expresiones no hemos completado los índices de los vectores con la operación de módulo que es necesaria para no salirse del intervalo permitido a los índices ($[0, C_\mathrm{sk} - 1]$), es decir, en donde dice $\mathtt{X}(i)$ lo más correcto sería que pusiese $\mathtt{X}((i + C_\mathrm{sk}) \text{ mód } C_\mathrm{sk})$ y $\mathrm{Menor}(a,b)$ se define como:

$$\mathrm{Menor}(a,b) = \begin{cases} 1 & \text{si } a < b \\ 0 & \text{si } a \geq b \end{cases} \tag{2.26}$$

El valor $\bar{L}(j,h)$ es el número medio de líneas de este patrón de acceso que caen en el conjunto al que pertenece la línea asociada a la región cuyo final se ha detectado entre el acceso a la misma en la presente iteración y la siguiente, en la cual se habrá desplazado una palabra. Esta función depende de la posición relativa a la línea h de la palabra final de la región en la primera iteración. Obsérvese que no tiene sentido calcular $\bar{L}(j, L_\mathrm{s} - 1)$ ya que en la iteración siguiente, dado que la región comienza una palabra después, también finaliza

una palabra después, con lo que la palabra final de la región pasa a pertenecer a otro conjunto.

Por otro lado, los valores $\bar{F}(j,i)$ y $\bar{C}(j,i)$ nos dan respectivamente el aporte en líneas de las regiones que se iniciasen en la última palabra de la línea considerada o que finalizasen en la última palabra de la línea anterior. Requieren una consideración especial, pues dependiendo del orden de los accesos de estas regiones con respecto a la estudiada, aportarán un número de línea diferente a $\bar{L}(j,h)$. Como ejemplo explicaremos la ecuación (2.24). El término mín$\{1,\texttt{CV}(i)\}$ multiplicando el resto de la fórmula hace que adopte el valor cero si $\texttt{CV}(i) = 0$, y que adopte el valor por el que se multiplica siempre que $\texttt{CV}(i) \neq 0$. El resto de la expresión nos da el número medio de líneas que las regiones que comenzasen en la posición i aportarían a las interferencias con las que acaban en la posición j considerando que i es la última posición de una línea de este conjunto. Debemos tener en cuenta que estas $\texttt{CV}(i)$ regiones aportarán líneas de interferencia sólo en sus accesos de la primera iteración, pues al comenzar en la última palabra de la línea, en la siguiente iteración comenzarán en la siguiente línea, y por tanto en el siguiente conjunto. Si numeramos de 0 a $\texttt{FV}(j) - 1$ las regiones cuya última palabra ocupa la posición j en orden de acceso, para la región h-ésima este número de líneas sería:

$$L_{intf}(j,i,h) = \mathrm{C}V(i) - \mathrm{Menor}(\mathrm{T}(i + T_\mathrm{R} - 1), \mathrm{T}(j)) - h \tag{2.27}$$

La explicación de esta ecuación es que en el momento de acceder la región h ya se ha accedido a h de las regiones que comienzan o terminan en cualquier punto de la caché, puesto que en este tipo de acceso no es posible repetir un acceso al mismo punto de la caché en tanto no se hayan referenciado todos los restantes puntos afectados por él. Por otra parte, si las regiones asociadas al punto i con el que estamos comparando son accedidas antes que las que estamos considerando ($\mathrm{T}(i+T_\mathrm{R}-1) < \mathrm{T}(j)$), habrá una región más accedida previamente (véase figura 2.8). El motivo de sumar $T_\mathrm{R}-1$ a i para obtener su momento de acceso es que, como se recordará, $\texttt{T}(j)$ contiene mín$\{i/F_i = j\}$, mientras nosotros estamos considerando las regiones que comienzan en el punto i, por lo que es necesario sumar dicha cantidad para calcular el punto donde finalizan.

Es directo comprobar que la media de estos valores es:

$$\sum_{h=0}^{\mathrm{F}V(j)-1} \frac{L_{intf}(j,i,h)}{\mathrm{F}V(j)} = \mathrm{C}V(i) - \mathrm{Menor}(\mathrm{T}(i + T_\mathrm{R} - 1), \mathrm{T}(j)) - \frac{\mathrm{F}V(j) - 1}{2} \tag{2.28}$$

La ecuación (2.25) se obtiene siguiendo un razonamiento análogo con respecto al final de las regiones.

La construcción del vector de área de auto-interferencia se realiza de la siguiente forma tras el cálculo de cada valor:

$$S'_{\mathrm{ra}}(N_{\mathrm{R}}, T_{\mathrm{R}}, L_{\mathrm{R}}) = \frac{\sum_{j=0}^{C_{\mathrm{sk}}-1} \mathrm{F}V(j) \sum_{h=0}^{L_{\mathrm{s}}-2} S_{\mathrm{s}}(\bar{L}(j,h)C_{\mathrm{sk}})}{N_{\mathrm{R}}(L_{\mathrm{s}}-1)} \tag{2.29}$$

La idea es promediar para cada conjunto de regiones que finalizan en un punto j los $L_{\mathrm{s}} - 1$ vectores de área de auto-interferencia que pueden tener en el tipo de acceso que estamos estudiando. Dichos vectores se multiplican por el número de regiones a que afectan para ser promediados.

2.4. Simplificaciones en el modelado de cachés de correspondencia directa

El proceso de desarrollo de los modelos presentados en este trabajo comenzó por abordar el estudio del comportamiento de las cachés de mapeado directo, dando lugar a las publicaciones [22] y [19]. Posteriormente se realizó una generalización [23] para contemplar cualquier caché con un nivel de asociatividad arbitrario y una estrategia de reemplazo LRU, que es, como ya se ha mencionado, la más frecuente con diferencia en los sistemas reales. La consideración de un nivel de asociatividad $K = 1$ permite realizar una serie de simplificaciones sobre las fórmulas del modelo que conllevan una importante reducción de los tiempos de computación. Por este motivo se ha considerado la inclusión de ciertas fórmulas específicas para cachés de mapeado directo que pueden ser de interés para el modelado de algoritmos que den lugar a tiempos de computación elevados.

Un primer aspecto a tener en cuenta es que un vector de área de interferencia para una caché de correspondencia directa constará únicamente de dos elementos: el primero representará la proporción de líneas de la caché que han sido afectadas por el acceso asociado al vector, mientras que el segundo dará la porción de líneas que no han sido afectadas. Dado que la suma de los componentes de un vector de área siempre ha de dar 1 (el 100 % de los conjuntos de la caché), podemos simplificar los vectores de área representándolos de forma abreviada mediante la proporción s de conjuntos de la caché involucrados en el acceso modelado por el vector, es decir, el primer componente del vector.

A partir de esta simplificación puede deducirse una forma simple de efectuar la unión de porciones de área: bastará con sumarlas como probabilidades independientes, es decir, dadas dos porciones s_1 y s_2, puede estimarse su unión como:

$$s_1 \cup s_2 = s_1 + s_2 - s_1 s_2 \tag{2.30}$$

También puede simplificarse el cálculo de los vectores de área (ahora porciones de área) asociados a varios patrones de acceso, como veremos en el siguiente apartado.

2.4.1. Patrones de acceso

Repasando la sección 2.3 puede apreciarse que el acceso secuencial y el correspondiente a una probabilidad uniforme de acceso por línea tienen un modelado que no se basa en el de ningún otro patrón. Las fórmulas para el modelado de los restantes patrones se generan a partir de las de estos dos patrones básicos, aplicándoles probabilidades obtenidas de forma totalmente analítica (accesos con probabilidad uniforme de acceso por grupos de elementos) o mixta (acceso a regiones separadas por una distancia constante). La consideración de una caché de correspondencia directa permite simplificar precisamente las expresión del modelado de estos dos patrones básicos, afectando así también el modelado de los restantes.

Acceso secuencial

En el apartado 2.3.1 se vió que el acceso a n palabras trae por término medio $n + L_\mathrm{s} - 1$ palabras a la caché debido a las palabras adicionales que se pueden encontrar en la primera y en la última de las líneas afectadas por el acceso. Dado que en la caché de mapeado directo representamos el área afectada de alguna manera por un acceso simplemente por la porción que ésta representa sobre el total de la misma, la expresión (2.4) que proporciona $S_\mathrm{s}(n)$ se simplifica de la siguiente forma:

$$S_\mathrm{s}(n) = \mathrm{mín}\left\{1, \frac{n + L_\mathrm{s} - 1}{C_\mathrm{s}}\right\} \tag{2.31}$$

Acceso a líneas con una probabilidad uniforme de referencia

Si consideramos un acceso a n palabras en el que cada línea tiene una probabilidad uniforme P_l de ser accedida, tendremos $\lfloor n/C_\mathrm{s} \rfloor$ capas del tamaño

de la caché , más una última de $n - \lfloor n/C_\mathrm{s} \rfloor C_\mathrm{s}$ elementos. Las líneas en cada una tendrán una P_l de haber sido accedidas. Así pues, siguiendo con el uso de la distribución binomial para estimar la solución, todas las líneas tendrán una probabilidad al menos $1 - (1 - P_\mathrm{l})^{\lfloor n/C_\mathrm{s} \rfloor}$ de haber sido accedidas, y una proporción $(n/C_\mathrm{s}) - \lfloor n/C_\mathrm{s} \rfloor$ de ellas tendrán una probabilidad adicional de acceso P_l. De esta forma, en lugar de utilizar la expresión (2.5), podemos estimar $S_\mathrm{l}(n, P_\mathrm{l})$ como:

$$\begin{aligned} S_\mathrm{l}(n, P_\mathrm{l}) &= (1 - (1 - P_\mathrm{l})^{\lfloor n/C_\mathrm{s} \rfloor}) \cup \left(\frac{n}{C_\mathrm{s}} - \left\lfloor \frac{n}{C_\mathrm{s}} \right\rfloor \right) P_\mathrm{l} \\ &= 1 + ((n/C_\mathrm{s} - \lfloor n/C_\mathrm{s} \rfloor) P_l - 1)(1 - P_l)^{\lfloor n/C_\mathrm{s} \rfloor} \end{aligned} \tag{2.32}$$

Capítulo 3

Modelado de cachés asociativas por conjuntos para computaciones irregulares

En este capítulo ilustraremos la aplicación de la estrategia de modelado analítico cuyas bases asentamos en el capítulo anterior aplicándolo a diversos códigos de álgebra matricial dispersa que contienen patrones de acceso irregulares. Estos patrones proceden del uso de formatos de almacenamiento comprimido para estas matrices, los cuales permiten ahorrar operaciones y espacio de memoria pero que obligan a realizar accesos indirectos. En estos códigos emplearemos siempre el formato de almacenamiento CRS [7] que consta de tres vectores: `A` contiene las entradas de la matriz dispersa, `C` almacena la columna de cada entrada, y `R` indica en qué punto de `A` y `C` comienza una fila de la matriz dispersa. Además, efectuaremos el modelado para una distribución uniforme de los elementos no nulos de la matriz. Tanto este método de almacenamiento como la consideración de una dispersión uniforme son utilizadas extensivamente en la bibliografía relacionada [46], [49], [38].

La tabla 3.1 muestra los parámetros que describen las matrices que se utilizarán y algunos valores derivados. De entre ellos cabe destacar que p_{n} también puede considerarse como la densidad de la matriz. Por otra parte, la distribución uniforme de las entradas en la matriz dispersa nos permite considerar que el número de entradas en un grupo de L_{s} posiciones, tantas como elementos puede contener una línea, pertenece a una binomial $\mathrm{B}(L_{\mathrm{s}}, p_{\mathrm{n}})$. De esta forma, la probabilidad de que haya al menos una entrada en dicho grupo es $p = 1 - (1 - p_{\mathrm{n}})^{L_{\mathrm{s}}}$.

El modelado de cada algoritmo irá seguido de una validación realizada me-

M	Número de filas
N	Número de columnas
N_{nz}	Número de valores no nulos
β	Media de entradas por fila (N_{nz}/M)
p_{n}	Probabilidad de que una posición de la matriz dispersa contenga un no nulo (β/N)
p	Probabilidad de que haya al menos una entrada en L_{s} posiciones de la matriz dispersa

Cuadro 3.1: Parámetros de la matriz.

diante simulaciones efectuadas sobre matrices sintéticas con una distribución uniforme de las entradas. Las trazas para las simulaciones se han obtenido ejecutando los algoritmos tras reemplazar los accesos originales por llamadas a funciones que calculan la posición a acceder y la graban en disco. Las trazas fueron procesadas inicialmente usando el simulador de caché dineroIII, integrado en el WARTS toolset [32]. Cuando el tamaño de las trazas hizo muy costosa computacionalmente la simulación efectuada por esta herramienta, se hizo necesario el desarrollo de un simulador simplificado cuya validez fue contrastada comparando sus resultados con los de dineroIII. Otro inconveniente que surgió fue la aparición de trazas de tal tamaño que generaban problemas de almacenamiento en disco. En estos casos se alimentaron las direcciones de acceso generadas por la simulación de la ejecución del algoritmo directamente al simulador simplificado. Esta política, por otra parte, redujo aún más los tiempos de simulación, al evitarse las latencias de acceso a disco, si bien cada simulación de una configuración de caché diferente requería también la simulación de la ejecución del algoritmo. Finalmente, tras la validación de cada modelo se hará un breve análisis del comportamiento del algoritmo en función de sus parámetros a fin de ilustrar una de las utilidades del modelado; la comprensión de la relación entre dichos parámetros y el rendimiento de caché para un algoritmo dado.

En cuanto a los algoritmos analizados, que trataremos en orden creciente de complejidad, son los siguientes:

- El producto de una matriz dispersa por un vector.
- El producto de una matriz dispersa por una matriz densa. Si se utiliza el almacenamiento CRS para la matriz densa, como hemos indicado, hay tres anidamientos posibles de los bucles implicados en este algoritmo. Modelaremos las tres posibilidades.

- La trasposición de una matriz dispersa en la que tanto la matriz origen como la destino se almacenarán en formato CRS.
- Una versión altamente optimizada del producto matriz dispersa-matriz densa que incluye *blocking* tanto a nivel de caché como de registros.

3.1. Un procedimiento sistemático

Nuestra estrategia para el modelado será construir una expresión que proporcione el número de fallos F_{X} sobre cada estructura de datos `X`, que en los algoritmos que estudiaremos será un vector o una matriz. En el caso de que haya accesos a dicha estructura en varios puntos del algoritmo, se generará una expresión para obtener el número de fallos que pueden atribuirse a cada uno de los puntos en donde puede ser referenciada la estructura. Esta expresión constará de un término para cada posible distancia en el reuso por parte de la referencia que se esté modelando de una línea de la caché que contenga datos de la estructura `X`. El término será el producto del número de accesos sobre la estructura de datos considerada que corresponden a esta distancia de reuso por la probabilidad de que cada uno de esos accesos resulte en un fallo. Esta última vendrá dada, cuando la distancia sea constante, por el primer componente del vector de área de interferencia total $S_{\mathrm{int\ X}}$ asociado a los accesos que se dan entre los reusos de la línea. En general, dicho vector tendrá la forma:

$$S_{\mathrm{int\ X}} = S_{\mathrm{auto\ X}} \cup S_{\mathrm{cruzada\ X}} \tag{3.1}$$

es decir, será la unión del vector de área de auto-interferencia generada por la estructura misma con el vector de área de interferencia cruzada generada por los accesos a las restantes estructuras de datos durante el periodo de ejecución considerada. En ocasiones, como por ejemplo, durante un acceso secuencial puro, el vector de área de auto-interferencia será inexistente.

Cuando la distancia, medida siempre en términos de iteraciones de bucles, entre dos accesos consecutivos a una línea no sea constante, será necesario calcular estos vectores de área para cada una de las distancias posibles. En estos casos tendremos expresiones del tipo:

$$S_{\mathrm{int\ X}}(i) = S_{\mathrm{auto\ X}}(i) \cup S_{\mathrm{cruzada\ X}}(i) \tag{3.2}$$

para cada una de ellas. Los vectores de área así obtenidos se promediarán en fórmulas que tendrán en cuenta la probabilidad de que la distancia entre dos reusos consecutivos sea i, para así calcular la probabilidad media $P_{\mathrm{ac\ X}}$ de que haya un acierto en un acceso a la estructura de datos estudiada.

```
DO I=1, M
  REG = 0
  DO J=R(I), R(I+1)-1
    REG = REG + A(J) * X(C(J))
  ENDDO
  D(I) = REG
ENDDO
```

Figura 3.1: Producto matriz dispersa-vector.

Cuando una estructura de datos puede resultar accedida en más de un bucle del algoritmo analizado, distinguiremos a la hora de obtener la expresión del número de fallos en cada bucle entre dos tipos de vectores de área de interferencia. Uno será $S_{\text{int X}}$, que estará asociado a los accesos realizados desde el último acceso a una línea considerada dentro del bucle estudiado. El segundo, $S_{\text{int ext X}}$, corresponderá a todas las referencias efectuadas desde el último acceso a dicha línea en el bucle precedente donde era referenciable. La excepción será, obviamente, el primer bucle, donde no existirá este segundo vector.

3.2. Producto matriz dispersa-vector

El código de este algoritmo de álgebra matricial dispersa se muestra en la Figura 3.1. En él puede apreciarse que todos los vectores excepto `X` presentan un acceso puramente secuencial. Dos de ellos son accedidos en cada iteración del bucle más interno en `J` (`A` y `C`), mientras que los otros dos son accedidos una vez por iteración del bucle en `I`: los vectores `R` y `D`. Esto último será cierto si suponemos que el compilador es capaz de considerar que el valor de `R(I+1)` en una iteración dada del bucle externo es el valor de `R(I)` en la siguiente, lo cual quiere decir que en cada iteración realmente sólo es preciso acceder a `R(I+1)`, exceptuando la primera. Haremos un modelado por separado para cada una de estas dos clases de vectores y el vector `X`, que es el que experimenta un acceso irregular debido al acceso indirecto a través de los valores almacenados en el vector `C`.

3.2.1. Vectores `A` y `C`

Al comenzar el análisis de un código asumiremos que no hay ninguna porción de las estructuras de datos que emplea en la caché al iniciarse su ejecución. Así pues, al acceder secuencialmente a un vector habrá una probabilidad de fallo total en el primer acceso a cada línea de las que lo constituyen (fallos intrínsecos, que estimaremos como $N_{\mathrm{nz}}/L_{\mathrm{s}}$). Para las restantes, tal y como se explicó en el capítulo anterior, estimaremos la probabilidad de fallo como el elemento 0 del vector de área de las interferencias generadas entre dos accesos consecutivos a la línea.

Si tomamos el vector `A`, entre cada dos accesos consecutivos a él tenemos un acceso a `C` y otro a `X`. Dado que se trata de un solo acceso a cada vector, a la caché sólo se trae una línea de cada uno, con lo que el acceso secuencial es adecuado para modelar este tipo de referencia. El vector de área para `A` sería:

$$S_{\mathrm{int\ A}} = S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(1) \tag{3.3}$$

y la correspondiente probabilidad de fallo en el acceso a las últimas $L_{\mathrm{s}} - 1$ palabras de la línea $S_{\mathrm{int\ A_0}}$. Por tanto el número de fallos sobre el vector `A` se estimaría como:

$$F_{\mathrm{A}} = \frac{N_{\mathrm{nz}}}{L_{\mathrm{s}}}(1 + (L_{\mathrm{s}} - 1)S_{\mathrm{int\ A_0}}) \tag{3.4}$$

El cálculo de $S_{\mathrm{int\ A}}$ podría refinarse teniendo en cuenta que aproximadamente cada β accesos finaliza el procesamiento de una fila de la matriz dispersa, con lo que se sale del bucle interno del algoritmo y se producen dos accesos más: uno a `D` y otro a `R`. No obstante, el aumento de precisión obtenido es muy pequeño.

El vector `C` sigue exactamente el mismo patrón de acceso que el vector `A`, distinguiéndose sólo en que está formado por enteros en lugar de por valores en punto flotante. Dado que estamos usando como unidad de medida de la memoria el tamaño de un número en punto flotante, será preciso usar el parámetro r introducido en la sección 2.1 para considerar los valores enteros:

$$F_{\mathrm{C}} = \frac{N_{\mathrm{nz}} r}{L_{\mathrm{s}}}(1 + (L_{\mathrm{s}}/r - 1)S_{\mathrm{int\ C_0}}) \tag{3.5}$$

donde $S_{\mathrm{int\ C}} = S_{\mathrm{int\ A}}$.

3.2.2. Vectores `R` y `D`

Estamos en una situación similar a la anterior: ambos vectores tienen el mismo patrón de acceso , distinguiéndose únicamente en que uno está constituido por enteros y el otro por valores en punto flotante. Dado que el mecanismo de paso de un tipo de vector a otro ya se ha ilustrado en el apartado anterior, aquí sólo efectuaremos el modelado para el vector `D`. El vector de área de interferencia será:

$$S_{\text{int D}} = S_{\text{s}}(\beta) \cup S_{\text{s}}(\beta r) \cup S_{\text{l}}(N, p) \cup S_{\text{s}}(1) \tag{3.6}$$

puesto que entre dos accesos consecutivos al vector `D` tendremos una iteración completa del bucle más interno, donde se accede por término medio a β elementos del vector `A` y otros tantos del vector `C` (escalados mediante r al ser enteros) secuencialmente, y habrá un acceso al vector `X` con una probabilidad uniforme de acceso por línea p. Además habrá un acceso al vector `R`.

El número de fallos sobre estos vectores, F_{D} y F_{R}, se calcula de forma análoga a la expresada en (3.4) y (3.5), respectivamente, con la salvedad de que en lugar de constar de N_{nz} elementos, contienen M.

3.2.3. Vector `X`

El vector `X` experimenta una serie de accesos indirectos dependientes de la localización de las entradas en la matriz dispersa, dado que se direcciona a partir del valor en `C(J)`. Dado que todas sus líneas tienen la misma probabilidad de ser accedidas durante el producto escalar por una fila de la matriz dispersa, en virtud de la distribución uniforme de los elementos no nulos en la matriz, nos encontramos ante un acceso con probabilidad uniforme de acceso por línea.

El número de fallos en el acceso a este vector se calcula multiplicando el número de líneas referenciadas durante cada producto escalar por el número de filas de la matriz dispersa y la probabilidad de fallo en cada uno de estos accesos. El número de líneas diferentes accedidas durante el procesamiento de cada fila de la matriz dispersa viene dado por pN/L_{s}, ya que `X` cubre N/L_{s} líneas, y cada una tiene una probabilidad p de ser accedida durante el producto escalar con una fila de la matriz dispersa, ya que es la probabilidad de que haya al menos una entrada en algunas de las L_{s} posiciones de la fila de la matriz cuyas columnas corresponden a las posiciones de una línea del vector `X`.

La probabilidad de fallo se calcula como la opuesta a la probabilidad de acierto. Los aciertos tienen lugar cuando la línea accedida ha sido referenciada durante un producto escalar anterior y no ha sufrido ni auto-interferencias ni interferencias cruzadas.

Vector de área de interferencia cruzada

Las interferencias cruzadas son generadas por los accesos a los restantes vectores, que presentan un acceso secuencial. El área de interferencia cruzada viene dada por el vector de área total asociado a los accesos a los vectores `A`, `C`, `R` y `D`. Este vector de área se calcula, como ya hemos visto, sumando los vectores de área individuales correspondientes a los accesos a cada vector del programa durante el periodo considerado. En nuestro caso estamos interesados en calcular el vector de área de interferencia cruzada tras i productos escalares:

$$S_{\text{cruzada X}}(i) = S_{\text{A}}(i) \cup S_{\text{C}}(i) \cup S_{\text{R}}(i) \cup S_{\text{D}}(i) \tag{3.7}$$

donde $S_{\text{V}}(i)$ es el vector de área correspondiente a los accesos a un vector `V` durante i productos escalares. Los cuatro vectores tienen un acceso secuencial, por lo que dichos valores se calculan como:

$$\begin{aligned} S_{\text{R}}(i) = &\quad S_{\text{s}}(ir) \quad S_{\text{D}}(i) = S_{\text{s}}(i) \\ S_{\text{A}}(i) = &\quad S_{\text{s}}(i\beta) \quad S_{\text{C}}(i) = S_{\text{s}}(i\beta r) \end{aligned} \tag{3.8}$$

Estos valores se obtienen considerando que en cada producto escalar se accede a un elemento de `D` y `R` y a β componentes de `A` y `C`. Se aplica el factor de escala r a los vectores compuestos por elementos enteros, como ya se hizo en los apartados anteriores.

Vector de área de auto-interferencia

En cuanto a la probabilidad de auto-interferencia, cada línea de `X` compite por término medio con $C(N)$ líneas del mismo vector (ver definición de $C(n)$ en (2.3)) en el mismo conjunto de la caché, y todas ellas tienen una probabilidad p de ser accedidas durante un producto escalar con una fila de la matriz dispersa. Como resultado, el número de líneas de `X` que pueden reemplazar a otra tras i productos escalares pertenece a una binomial $\text{B}(C(N), 1-(1-p)^i)$. Así pues, este patrón corresponde a un acceso con probabilidad uniforme de referencia por línea $1-(1-p)^i$ a las líneas del vector `X` que pueden generar

auto-interferencias con otra. De ahí que el vector de área de auto-interferencia se calcule como:

$$S_{\text{auto X}} = S_{\text{l}}(C(N)C_{\text{sk}}, 1-(1-p)^i) = S^i_{\text{la}}(N,p) \tag{3.9}$$

Cálculo final del número de fallos sobre el vector `X`

La probabilidad de acierto en el primer acceso a cualquier línea de `X` durante el producto escalar de la j-ésima fila de la matriz dispersa por el vector `X` es:

$$P_{\text{ac X}}(j) = \sum_{i=1}^{j-1} p(1-p)^{i-1}(1 - S_{\text{int X}_0}(i)) \tag{3.10}$$

donde $p(1-p)^{i-1}$ es la probabilidad de que el último acceso a la línea haya tenido lugar hace i productos escalares y $S_{\text{int X}}(i)$ es el vector de área de interferencia generado por los accesos producidos durante esos i productos escalares. De acuerdo con la expresión (3.2), $S_{\text{int X}}(i)$ se calcula como la unión de los vectores de área de auto-interferencia y de interferencia cruzada (expresiones (3.7) y (3.9) respectivamente).

Finalmente, la probabilidad media de acierto se calcula como:

$$P_{\text{ac X}} = \frac{\sum_{j=1}^{M} P_{\text{ac X}}(j)}{M} \tag{3.11}$$

Debemos señalar que el modelo explicado sólo tiene en cuenta el primer acceso a cada línea de `X` en cada producto escalar. Los otros $N_{\text{nz}} - pMN/L_{\text{s}}$ accesos tienen una probabilidad muy baja de resultar en fallo, dado que referencian líneas accedidas en la iteración previa del bucle más interno. El vector de interferencia para ellos, $S_{\text{int J X}}$, sería idéntico a $S_{\text{int A}}$, puesto que entre dos accesos consecutivos a una línea en `X`, lo cual sólo puede ocurrir en iteraciones consecutivas, sólo se accede a dos palabras: una del vector `A` y otra de `C`. Por tanto, el número de fallos sobre `X` puede estimarse como:

$$F_{\text{X}} = pM\frac{N}{L_{\text{s}}}(1 - P_{\text{ac X}}) + \left(N_{\text{nz}} - pM\frac{N}{L_{\text{s}}}\right) S_{\text{int J X}_0} \tag{3.12}$$

3.2.4. Validación y análisis

La tabla 3.2 muestra la desviación Δ de las predicciones del modelo para el producto matriz dispersa-vector para algunas combinaciones de los parámetros de entrada. En ella, al igual que haremos en los restantes algoritmos,

N	N_{nz}	p_{n}	C_{s}	L_{s}	K	fallos medidos	fallos predichos	σ	Δ
1	10	1.0 %	2	4	1	8083.60	8093.58	3.04	0.12
1	10	1.0 %	2	4	2	7015.12	6952.84	6.15	-0.89
1	10	1.0 %	4	4	4	5782.86	5779.39	0.25	-0.06
1	10	1.0 %	8	4	1	6422.36	6405.88	2.36	-0.26
1	10	1.0 %	8	8	2	2898.52	2903.55	0.94	0.17
1	10	1.0 %	16	8	4	2876.00	2875.25	0.00	-0.03
1	100	10.0 %	1	8	1	50993.32	52220.85	6.78	2.41
1	100	10.0 %	16	4	2	50846.80	50943.57	0.62	0.19
1	100	10.0 %	32	8	2	25379.44	25390.34	0.03	0.04
10	100	0.1 %	8	8	2	70199.33	69978.14	2.18	-0.32
10	100	0.1 %	16	8	1	43925.64	44047.74	4.12	0.28
10	100	0.1 %	16	8	4	36986.65	37134.21	4.77	0.15
10	100	0.1 %	32	8	2	30440.30	30239.15	6.16	0.35
10	100	0.1 %	64	16	1	16553.36	16670.91	3.23	0.71
10	100	0.1 %	64	8	4	28751.22	28751.04	0.00	0.00

Cuadro 3.2: Desviación del modelo para el producto matriz dispersa-vector.

hemos considerado matrices cuadradas ($N = M$), y $r = 1$. Los tamaños C_{s} y L_{s} se representan, respectivamente, en Kpalabras y en palabras, mientras que N y N_{nz} están en miles. Cada valor de la tabla se ha obtenido a partir del número de fallos obtenido en 20 simulaciones en las que se han modificado los valores de las posiciones iniciales de las estructuras de datos, a fin de contemplar diversas posiciones relativas de las mismas en la caché. Estas variaciones dan lugar a diferentes números de fallos. La columna σ representa la desviación típica del número de fallos medidos en las distintas simulaciones expresado como tanto por cien del número medio de fallos medidos. La desviación del modelo Δ viene expresada como el porcentaje sobre la media aritmética del número de fallos medidos en las simulaciones de la diferencia entre el número de fallos que predice el modelo y dicho valor medio medido. En esta ocasión hemos incluido también en la tabla de ejemplos de validación el número medio de fallos medidos en las simulaciones así como el valor predicho por el modelo.

En el conjunto de combinaciones de los parámetros de entrada que se han verificado para este algoritmo (en este caso se comprobaron unas 300 combinaciones), el error medio cometido por el modelo ha sido el 0.72 % pese a que la gran mayoría de los errores son inferiores a esta media. El motivo es la existencia de combinaciones en las que en alguna simulación se ha disparado

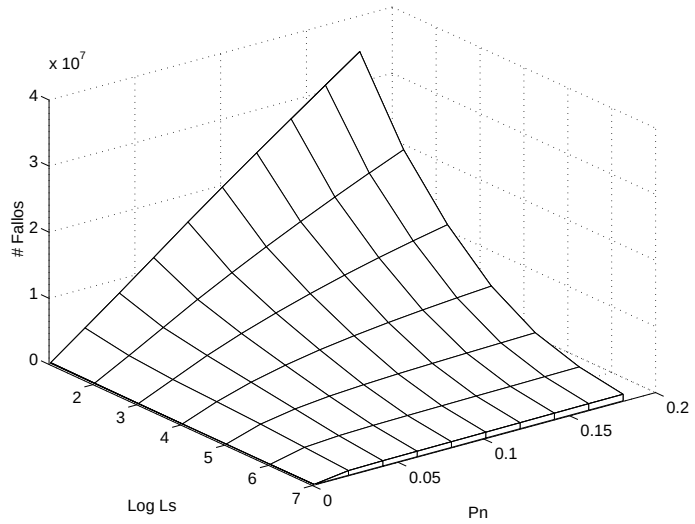

Figura 3.2: Número de fallos en una caché de 2K palabras asociativa por conjuntos de 4 vías durante el producto matriz dispersa-vector de una matriz $10K \times 10K$ en función de L_s y p_n.

el número de fallos al coincidir en la misma línea de la caché el inicio de los vectores `A`, y `C`, que se acceden a la par. Otro dato de interés es el valor medio de σ en nuestro conjunto de pruebas, que ha resultado ser el 3.15 %. Así pues, el error medio del modelo es sensiblemente inferior a la desviación típica media producida en las simulaciones.

Mediante el uso del modelo podemos ilustrar el comportamiento del algoritmo respecto a sus parámetros de entrada. Ejemplos de ello son las gráficas 3.2 a 3.4. En el caso de L_s mostramos el logaritmo base 2 del número de palabras la línea.

La Figura 3.2 muestra la relación del número de fallos con L_s y p_n en el producto matriz dispersa-vector. La evolución es aproximadamente lineal con respecto a p_n, ya que el número de accesos es directamente proporcional a N_nz y la mayoría de ellos siguen un patrón secuencial. Puede observarse cómo decrece significativamente el número de fallos al aumentar L_s debido a que los accesos a todos los vectores excepto `X` son secuenciales (ver Figura 3.1) y cuanto mayores sean las líneas mejor se explota de la localidad espacial. Sin embargo, cuando L_s es muy grande (> 64 palabras) y la matriz tiene una mayor densidad ($p_\mathrm{n} > 0{,}1$) el aumento de las probabilidades de interferencia sobre el vector `X` comienza a equilibrar las ventajas obtenidas del mejor uso de la localidad espacial mostrada por los otros vectores para $K = 1$. Este efecto, mostrado en la Figura 3.3, aumenta con el valor de p_n.

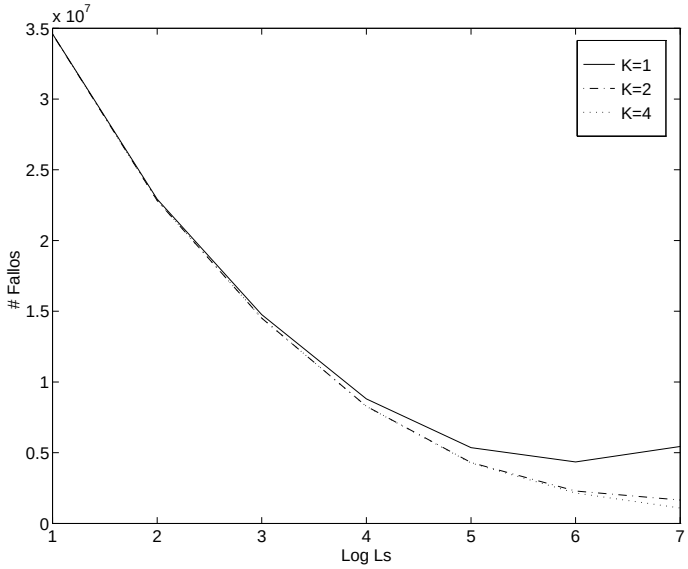

Figura 3.3: Número de fallos durante el producto matriz dispersa-vector de una matriz $10K \times 10K$ con $p_n = 0{,}18$ en una caché de 2K palabras en función de L_s para varios grados de asociatividad.

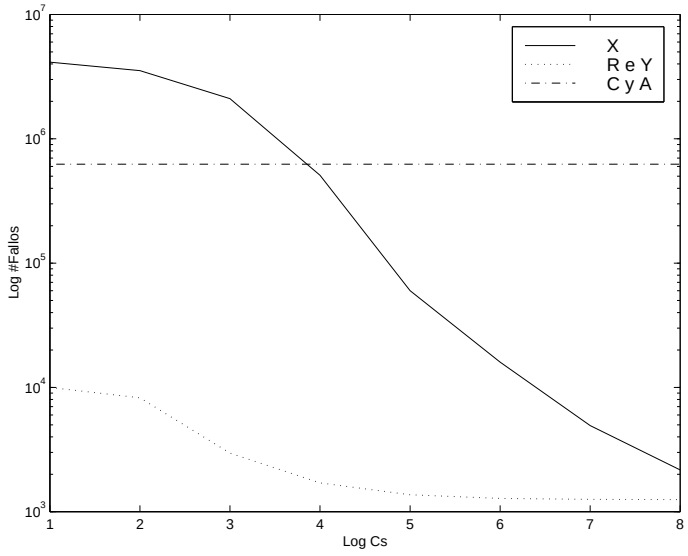

Figura 3.4: Número de fallos de cada vector durante el producto matriz dispersa-vector de una matriz $10K \times 10K$ con $p_n = 0{,}05$ en una caché de dos vías con $L_\mathrm{s} = 8$ palabras en función de C_s.

El comportamiento en función del tamaño de la caché podemos verlo en la figura 3.4, la cual además muestra como a través del uso del modelo podemos desglosar los fallos en los vectores a que corresponden. Se ha usado una escala logarítmica tanto para el número de fallos como para el tamaño de la caché, si bien para este último presentamos el logaritmo en base 2 del número de Kpalabras de que consta la caché, cosa que siempre haremos. Dado que la lectura de los vectores `A` y `C` es secuencial, el aumento del tamaño de la caché apenas tiene influencia sobre él, simplemente contribuye a reducir las interferencias cruzadas que puedan sufrir, que son pocas. Algo similar ocurre con `R` e `Y`, si bien al tener un número mucho menor de fallos y una probabilidad sustancialmente mayor de interferencia, el efecto es más acusado en la gráfica. Por último, el vector `X` es el que experimenta una reducción más fuerte de la tasa de fallos con el incremento de C_s. Por un lado, al pasar de 8 a 16 Kpalabras desaparece por completo la probabilidad de auto-interferencia en su acceso. Por otro lado, la probabilidad de interferencia cruzada que puede sufrir, al ser sustancialmente mayor que la que presentan los restantes vectores, experimenta también una reducción mucho mayor al aumentar la capacidad de la caché.

3.3. Producto matriz dispersa-matriz densa: orden JIK

Hay tres versiones básicas para el código del producto matriz dispersa-matriz densa con almacenamiento CRS, dependiendo del orden en el que se aniden los bucles: JIK, IJK e IKJ, donde la primera letra corresponde al bucle más externo y la última al más interno. Los elementos de la matriz dispersa, la matriz densa `B` y la matriz producto `D` se referencian mediante `(I,K)`, `(K,J)` y `(I,J)`, respectivamente. El código para la primera versión lo encontramos en la figura 3.5

En el orden JIK, se multiplica la matriz dispersa por una columna de la matriz densa `B` en cada iteración del bucle en `J`. Así pues, se trata de una repetición del producto matriz dispersa-vector H veces, siendo H el número de columnas de la matriz densa. En cada iteración de dicho bucle se usa una columna distinta de `B` y de `D`, con lo que multiplicar por H el número de fallos que se generarían en el producto matriz dispersa-vector correspondiente a una iteración es una buena aproximación.

Sin embargo, esta aproximación no sería correcta para los vectores `A`, `C` y `R`, pues si bien en el primer producto se comportan como en el producto

```
DO J=1, H
  DO I=1, M
    REG=0.0
    DO K=R(I), R(I+1)-1
      REG=REG + A(K) * B(C(K), J)
    ENDDO
    D(I,J)= D(I,J)+REG
  ENDDO
ENDDO
```

Figura 3.5: Producto matriz dispersa-matriz densa con orden JIK.

matriz dispersa-vector, en los subsiguientes puede darse un acierto en el reuso de sus elementos, puesto que han sido accedidos previamente. Dedicamos los apartados siguientes a su modelado.

3.3.1. Vectores `A` y `C`

Una vez más, trataremos ambos vectores juntos por tener el mismo patrón de acceso. Dado que la diferencia con el producto matriz dispersa-vector radica en que puede haber probabilidad de acierto P_{ac} en el primer acceso a cada línea en el producto por una columna que no sea la primera, podemos reescribir la expresión (3.4) poniéndola en función de dicha probabilidad para obtener el número de fallos sobre el vector `A` durante el procesamiento de una columna:

$$F_{\text{col A}}(P_{\text{ac}}) = \frac{N_{\text{nz}}}{L_{\text{s}}}(P_{\text{ac}} + (L_{\text{s}} - 1)S_{\text{int A}_0}) \tag{3.13}$$

A lo largo del producto matriz dispersa-vector correspondiente a cada columna de la matriz `B`, se genera el siguiente vector de área de interferencia para el vector `A`:

$$\begin{aligned} S_{\text{int J A}} =& S_{\text{s}}(C(N_{\text{nz}})C_{\text{sk}}) \cup S_{\text{s}}(N_{\text{nz}}r) \cup S_{\text{s}}(Mr) \cup \\ & S_{\text{s}}(M) \cup S_{\text{l}}\left(2N, 1 - \frac{1-(1-p)^M}{pM}\right) \end{aligned} \tag{3.14}$$

donde los componentes corresponden, respectivamente, a las posibles auto-interferencias de un acceso a la totalidad de `A`, más las interferencias cruzadas producidas por las referencias a la totalidad de los vectores `C` y `R`, una columna de la matriz `D` y las porciones accedidas de dos columnas de la matriz `B`.

Este último valor podemos estimarlo empleando una distribución uniforme de probabilidad por línea sobre las $2N$ palabras de que constan las dos columnas. La probabilidad media será la media de las probabilidades de que una línea haya sido accedida a lo largo del procesamiento de $0, 1, \ldots, M-1$ filas de la matriz dispersa. Dado que la probabilidad de acceso por línea es p por cada fila, para i filas será $1-(1-p)^i$. La probabilidad en 3.14 es la media de este valor para $i = 0, 1, \ldots, M-1$.

El número de fallos sobre `A` se estimará en este caso como:

$$F_{\mathrm{A}} = F_{\mathrm{col\ A}}(1) + (H-1)F_{\mathrm{col\ A}}(S_{\mathrm{int\ J\ A_0}}) \tag{3.15}$$

El caso del vector `C` es análogo, teniendo en cuenta solamente la diferencia de escala por constar de elementos de tipo entero.

3.3.2. Vector `R`

Siguiendo un procedimiento similar, tendríamos que el número de fallos sobre `R` durante el procesamiento de una columna de la matriz densa en función de la probabilidad de acierto en el acceso al primer componente de cada línea de dicho vector, es:

$$F_{\mathrm{R\ col}}(P_{\mathrm{ac}}) = \frac{Mr}{L_{\mathrm{s}}}(P_{\mathrm{ac}} + (L_{\mathrm{s}}/r - 1)S_{\mathrm{int\ R_0}}) \tag{3.16}$$

donde $S_{\mathrm{int\ R}}$ adopta el mismo valor que $S_{\mathrm{int\ D}}$ en la sección 3.2.2. El vector de área de interferencia para `R` entre dos accesos consecutivos a la misma línea producidos durante el producto con dos columnas consecutivas de la matriz `B` sería:

$$\begin{aligned} S_{\mathrm{int\ J\ R}} =& S_{\mathrm{s}}(C(Mr)C_{\mathrm{sk}}) \cup S_{\mathrm{s}}(N_{\mathrm{nz}}r) \cup S_{\mathrm{s}}(N_{\mathrm{nz}}) \cup \\ & S_{\mathrm{s}}(M) \cup S_{\mathrm{l}}\left(2N, 1 - \frac{1-(1-p)^M}{pM}\right) \end{aligned} \tag{3.17}$$

es decir, adopta el valor de $S_{\mathrm{int\ J\ A}}$ reemplazando la auto-interferencia para el vector `A`, $S_{\mathrm{s}}(C(N_{\mathrm{nz}})C_{\mathrm{sk}})$, por la del vector `R` que estamos considerando actualmente, $S_{\mathrm{s}}(C(Mr)C_{\mathrm{sk}})$, y la interferencia externa asociada a `R`, $S_{\mathrm{s}}(Mr)$, por la correspondiente a `A`, $S_{\mathrm{s}}(N_{\mathrm{nz}})$.

La expresión del número de fallos sobre este vector se calcularía de forma análoga a (3.15).

N	N_{nz}	p_{n}	H	C_{s}	L_{s}	K	σ	Δ
1	10	1.0 %	100	2	4	1	23.69	-3.48
1	10	1.0 %	100	4	4	4	0.11	-0.01
1	10	1.0 %	100	8	4	1	1.50	0.17
1	10	1.0 %	100	8	8	2	1.41	0.08
1	10	1.0 %	100	16	8	4	3.88	0.40
1	10	1.0 %	100	64	8	2	29.50	6.89
2	20	0.5 %	200	1	8	1	1.11	1.00
2	20	0.5 %	200	16	4	2	1.98	0.04
2	20	0.5 %	200	32	8	4	3.53	-2.61
10	100	0.1 %	40	4	4	1	1.50	0.23
10	100	0.1 %	40	16	4	2	2.98	-0.01
10	100	0.1 %	40	16	8	1	3.13	-0.67
10	100	0.1 %	40	32	8	4	0.29	-0.06
10	100	0.1 %	40	64	16	2	1.09	0.09
10	100	0.1 %	1000	256	32	2	13.45	4.63

Cuadro 3.3: Desviación del modelo para el producto matriz dispersa-matriz densa con orden JIK.

3.3.3. Validación

La tabla 3.3 muestra los datos de validación para algunas de las posibles combinaciones de los parámetros para el producto matriz dispersa-matriz densa con orden JIK. Al igual que en la tabla 3.2 y las subsiguientes, N y N_{nz} están en miles; no obstante H estará siempre en unidades. Cabe destacar que del conjunto de combinaciones comprobadas se extrajo una tasa media de error para el modelo del 1.82 %, en tanto que la desviación típica media de las simulaciones resultó ser del 11.61 %. Puede apreciase que en este ordenamiento se dan grandes variaciones en el número de fallos en función de la posición relativa de los vectores implicados en la memoria. Esto se debe a la preponderancia de accesos de tipo secuencial repetidos a ciertas áreas de la memoria, en particular, los vectores que constituyen la matriz dispersa. Esta es también la causa de que para ciertas combinaciones de los parámetros la diferencia entre el número medio de fallos medidos en las simulaciones y el predicho sea grande. Cabría suponer que a medida que creciese el número de observaciones la media de fallos medidos iría convergiendo al valor del número de fallos predichos.

En el apartado 3.5.4 presentaremos un análisis comparativo del comportamiento del algoritmo para los tres ordenamientos.

```
DO I=1, M
  DO K=R(I), R(I+1)-1
    REG0=A(K)
    REG1=C(K)
    DO J=1, H
      D(I,J)=D(I,J)+REG0*B(REG1,J)
    ENDDO
  ENDDO
ENDDO
```

Figura 3.6: Producto matriz dispersa-matriz densa con orden IKJ.

3.4. Producto matriz dispersa-matriz densa: orden IKJ

Esta versión del algoritmo, cuyo código se muestra en la figura 3.6 no presenta las similaridades encontradas en el orden JIK con el producto matriz dispersa-vector, por lo que habremos de explicar el modelado de todas las estructuras de datos. Por otra parte, es la primera vez que encontramos accesos por filas, es decir, con una distancia constante entre los puntos accedidos.

3.4.1. Vector `R`

En esta ocasión el vector `R` es accedido secuencialmente y cada uno de sus elementos es referenciado una sola vez, con lo que puede calcularse el número de fallos sobre él como:

$$F_{\mathrm{R}} = \frac{Mr}{L_{\mathrm{s}}}(1 + (L_{\mathrm{s}}/r - 1)S_{\mathrm{int\ R_0}}) \tag{3.18}$$

calculándose el vector de área de interferencia como:

$$S_{\mathrm{int\ R}} = S_{\mathrm{s}}(\beta) \cup S_{\mathrm{s}}(\beta r) \cup S_{\mathrm{r}}(H, 1, M) \cup S_{\mathrm{l}}(NH, p) \tag{3.19}$$

cuyos componentes corresponden, respectivamente, a β elementos de los vectores `A` y `C`, que constituyen una fila de la matriz dispersa; una fila de la matriz de destino, y la totalidad de las líneas accedidas de `B`. El modelado del patrón de acceso a una fila de una matriz FORTRAN se realiza utilizando S_{r} (sección 2.3.5), puesto que ésta consta de una región de un elemento por cada columna de la matriz, estando separadas dos regiones consecutivas

por M palabras en la memoria, el tamaño de la columna, es decir, es un acceso a regiones separadas por una distancia constante. Dado que la matriz `B` está formada por NH elementos y cada línea tiene una probabilidad p de ser accedida durante el procesamiento de una fila de la matriz dispersa, su vector de área de interferencia cruzada debe ser $S_{\mathrm{l}}(NH, p)$.

3.4.2. Vectores `A` y `C`

Al igual que sucede con el vector `R`, los vectores `A` y `C` son accedidos secuencialmente y cada uno de sus elementos es referenciado una sola vez. Podemos usar las expresiones (3.4) y (3.5) para derivar el número de fallos para ellos, tomando en esta ocasión $S_{\mathrm{int\ A}}$ como:

$$S_{\mathrm{int\ A}} = S_{\mathrm{r}}(H, 1, M) \cup S_{\mathrm{r}}(H, 1, N) \cup S_{\mathrm{s}}(1) \tag{3.20}$$

donde los componentes se corresponden con el acceso a una fila de las matrices `D` y `B`, efectuado en el interior del bucle en `J`, y a un acceso al otro de los dos vectores (`C` para `A` y viceversa).

3.4.3. Matriz `D`

En la matriz `D` tenemos un acceso por filas efectuado para cada entrada de la matriz dispersa. Cada grupo de L_{s} filas de esta matriz será accedido si hay alguna entrada en las L_{s} filas correspondientes de la matriz dispersa, es decir, con probabilidad $1 - (1 - p_{\mathrm{n}})^{NL_{\mathrm{s}}}$. Estas filas cubren un total de H líneas en las que el primer acceso será un fallo intrínseco. En lo tocante a los $\beta L_{\mathrm{s}} - 1$ accesos restantes que tendrían de media, podríamos dividirlos en dos grupos teniendo en cuenta que las β entradas que hay en cada fila de la matriz dispersa por término medio producen accesos a pN/L_{s} líneas de cada columna de la matriz `B`.

- En cada fila de la matriz hay unas $\beta - pN/L_{\mathrm{s}}$ entradas que producen accesos a las líneas de una fila de la matriz densa que han sido accedidas en la iteración previa del bucle en `K`. Por tanto el vector de área de interferencia para la matriz `D`, $S_{\mathrm{int\ np\ D}}$, en los accesos producidos por estas entradas sería:

  $$S_{\mathrm{int\ np\ D}} = S_{\mathrm{ra}}(H, 1, M) \cup S_{\mathrm{r}}(H, 1, N) \cup S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(1) \tag{3.21}$$

  puesto que a la auto-interferencia del acceso a una fila de dicha matriz deberemos añadirle los vectores de área de interferencia cruzada del

acceso a las líneas de una fila de `B` y los accesos a un elemento de los vectores `A` y `C`. Recuérdese que S_{ra} (ver expresión (2.22) proporciona el vector de área de auto-interferencia en un acceso a regiones separadas por una distancia constante, que es precisamente el correspondiente al acceso a una fila en una matriz en FORTRAN, al almacenar este lenguaje las matrices por columnas.

- Las pN/L_{s} entradas restantes serán las primeras en su fila en acceder al grupo de líneas de `B` que constituyen una fila. Desde el punto de vista del acceso a la fila de la matriz `D`, esto querrá decir que entre el acceso anterior a la j-ésima línea de dicha fila y el actual, se habrá accedido a $H-j$ líneas procedentes de la fila anteriormente procesada de la matriz `B` y unas j de la actual, además de la propia fila de `D` y los elementos de los vectores `A` y `C`. Por tanto parece razonable calcular el vector de interferencia $S_{\text{int p D}}$ en estos accesos como:

$$\begin{aligned} S_{\text{int p D}} =& S_{\text{ra}}(H,1,M) \cup S_{\text{s}}(1) \cup S_{\text{s}}(1) \cup \\ & \frac{\sum_{j=1}^{H/2}(S_{\text{r}}(j,1,N) \cup S_{\text{r}}(H-j,1,N))}{H/2} \end{aligned} \tag{3.22}$$

De esta forma el número de fallos sobre la matriz `D` se estimaría como:

$$\begin{aligned} F_{\text{D}} =& \frac{M}{L_{\text{s}}}(1-(1-p_{\text{n}})^{NL_{\text{s}}}) \\ & (H + (\beta L_{\text{s}} - Np) S_{\text{int np D}_0} + (Np-1) S_{\text{int p D}_0}) \end{aligned} \tag{3.23}$$

3.4.4. Matriz `B`

Al igual que ocurría en el producto matriz dispersa-vector, tendremos $MN/L_{\text{s}}p$ entradas que serán la primera en su fila en generar accesos a una fila dada de `B`. Las restantes $N_{\text{nz}} - MN/L_{\text{s}}p$ estarán accediendo a una fila ya referenciada en la iteración anterior del bucle en `K`, con lo que el vector de área de interferencia para la matriz `B` en ellas será:

$$S_{\text{int np B}} = S_{\text{ra}}(H,1,N) \cup S_{\text{r}}(H,1,M) \cup S_{\text{s}}(1) \cup S_{\text{s}}(1) \tag{3.24}$$

que tiene la forma de (3.21) intercambiando el vector de área de auto-interferencia de una fila de `D` por el de una fila de `B` y el de interferencia cruzada de una fila de `B` por el de una de `D`.

En lo tocante al primer acceso a una fila de `B` durante el procesamiento de una fila de la matriz dispersa, usaremos la aproximación vista en el producto

matriz dispersa-vector: calcularemos la probabilidad de acierto en función del número de filas procesadas desde el último acceso a la que estamos estudiando. El vector de área de interferencia cruzada tras el procesamiento de i filas de la matriz dispersa será:

$$S_{\text{cruzada B}}(i) = S_{\text{s}}(i\beta) \cup S_{\text{s}}(i\beta r) \cup S_{\text{s}}(ir) \cup S_{\text{r}}(H, i, M) \tag{3.25}$$

cuyos miembros corresponden al acceso a β elementos de los vectores `A` y `C` y uno de `R` en cada fila, estando estos dos últimos constituidos por enteros; y al procesamiento de i filas de la matriz `D`. El vector completo de interferencia será:

$$S_{\text{int B}}(i) = S^{i}_{\text{la}}(NH, p) \cup S_{\text{cruzada B}}(i) \tag{3.26}$$

puesto que la auto-interferencia puede darse por el acceso a una cualquiera de las líneas de la matriz, de NH elementos, teniendo cada una una probabilidad de haber sido accedida a lo largo de i procesamientos de $1 - (1 - p)^i$. La expresión de la probabilidad de fallo durante el procesamiento de la j-ésima fila es análoga a la expresión (3.10). También el número de fallos se calcula de una forma muy similar a la del vector `X` en el producto matriz dispersa-vector:

$$F_{\text{B}} = H(pMN/L_{\text{s}}(1 - P_{\text{ac B}}) + (N_{\text{nz}} - pMN/L_{\text{s}})\, S_{\text{int np B}_0}) \tag{3.27}$$

donde $P_{\text{ac B}}$ es la media aritmética de los $P_{\text{ac B}}(j)$ para $j = 1, \ldots, M$, como en (3.11).

3.4.5. Validación

La tabla 3.4 contiene los datos de validación del modelo para algunas de las posibles combinaciones de los parámetros para el ordenamiento IKJ del producto matriz dispersa-matriz densa. En esta ocasión, del conjunto de combinaciones para las que se efectuaron simulaciones se obtuvo una tasa media de error en las predicciones del modelo del 0.62 %, mientras que la desviación típica media de las simulaciones resultó ser del 1.55 %.

3.5. Producto matriz dispersa-matriz densa: orden IJK

La última versión del algoritmo, ilustrada por el código de la figura 3.7, accede por filas la matriz `D` al igual que el orden IKJ, pero referencia la matriz densa `B` por columnas. Puede observarse que aunque el acceso a `R` se encuentra

N	N_{nz}	p_{n}	H	C_{s}	L_{s}	K	σ	Δ
1	10	1.0 %	100	2	4	1	0.30	-0.13
1	10	1.0 %	100	4	4	4	0.17	-0.10
1	10	1.0 %	100	8	4	1	0.29	-0.04
1	10	1.0 %	100	8	8	2	0.23	-0.03
1	10	1.0 %	100	16	8	4	0.38	-0.27
1	10	1.0 %	100	64	8	2	1.11	-0.31
2	20	0.5 %	200	1	8	1	0.00	0.00
2	20	0.5 %	200	16	4	2	0.10	-0.04
2	20	0.5 %	200	32	8	4	0.15	-0.12
10	100	0.1 %	40	4	4	1	0.06	0.01
10	100	0.1 %	40	16	4	2	0.05	-0.01
10	100	0.1 %	40	16	8	1	0.06	0.00
10	100	0.1 %	40	32	8	4	0.08	0.00
10	100	0.1 %	40	64	16	2	0.12	-0.04
10	100	0.1 %	1000	256	32	2	0.07	-0.37

Cuadro 3.4: Desviación del modelo para el producto matriz dispersa-matriz densa con orden IKJ.

```
DO I=1, M
  DO J=1, H
    REG=0
    DO K=R(I), R(I+1)-1
      REG=REG+A(K)*B(C(K),J)
    ENDDO
    D(I,J)=D(I,J)+REG
  ENDDO
ENDDO
```

Figura 3.7: Producto matriz dispersa-matriz densa con orden IJK.

dentro del bucle en `J`, en realidad sólo depende del bucle más externo, al igual que en el algoritmo anterior. Además las regiones accedidas durante una iteración del bucle más externo son las mismas que en el precedente, variando sólo el orden en el que se las accede. Por tanto el modelado del comportamiento del vector `R` es idéntico.

3.5.1. Vectores `A` y `C`

Al tener ambos vectores el mismo patrón, comentaremos sólo el modelado de `A`, como hemos venido haciendo. En este algoritmo se lee de forma secuencial este vector, repitiéndose H veces la lectura de cada grupo de β elementos correspondientes a una fila de la matriz dispersa antes de pasar al siguiente grupo. El vector de área de interferencia entre dos accesos consecutivos dentro del bucle en `K` es $S_{\text{int A}}$, calculado en (3.3), puesto que en ese periodo simplemente se accede a un elemento de `C` y otro de `B`. Este valor será aplicable a $L_{\text{s}} - 1$ de los elementos de cada línea. El acceso al primer elemento de una línea será un fallo intrínseco durante el procesado de la primera columna. En los restantes le corresponderá un vector de área de interferencia:

$$S_{\text{int J A}} = S_{\text{s}}(\beta r) \cup S_{\text{s}}(1) \cup S_{\text{l}}(N, p) \tag{3.28}$$

puesto que entre los accesos a los elementos del vector `A` pertenecientes a una fila de la matriz dispersa correspondientes al procesado de dos columnas se leen los β elementos asociados de `C`, uno de la matriz `D` y se realiza un acceso a una columna de la matriz `B` con probabilidad uniforme de acceso por línea p. Así pues, el número de fallos sobre el vector `A` se estimaría como:

$$F_{\text{A}} = \frac{N_{\text{nz}}}{L_{\text{s}}}(1 + (H-1)S_{\text{int J A}_0}) + \left(N_{\text{nz}} - \frac{N_{\text{nz}}}{L_{\text{s}}}\right) H S_{\text{int A}_0} \tag{3.29}$$

3.5.2. Matriz `D`

Esta matriz es accedida por filas, habiendo un fallo intrínseco en el primer acceso a cada línea de la misma, aproximándose el vector de área de interferencia para los restantes accesos mediante la expresión:

$$S_{\text{int np D}} = S'_{\text{ra}}(H, 1, M) \cup S_{\text{s}}(2\beta) \cup S_{\text{s}}(2\beta r) \cup S_{\text{l}}(NH, p) \tag{3.30}$$

que agrupa la auto-interferencia sobre la línea de `D` teniendo en cuenta que las líneas correspondientes a las columnas posteriores de la fila son accedidas con índice de fila i mientras que los de las columnas precedentes se acceden en la siguiente iteración, con índice $i+1$. Este tipo de vector de área de auto-interferencia se calcula mediante S'_{ra} (véase apartado 2.3.5). Incluye también los accesos a los elementos de los vectores `A` y `C` asociados a las líneas i e $i+1$, y realiza una aproximación de las porciones accedidas de la matriz `B` en el intervalo considerado. Estas porciones son los datos en las columnas con índice superior a la que se está considerando de `D` correspondientes a las filas direccionadas por las entradas de la fila i de la matriz dispersa, y

en las columnas con índice inferior asociadas a las filas direccionadas por las entradas de la siguiente fila. Así pues, el vector de área de interferencia cruzada puede estimarse como el correspondiente a una distribución uniforme de la probabilidad de acceso p sobre toda la matriz, dado que en cada fila de la matriz dispersa las entradas se distribuyen de forma que generan accesos a pN/L_{s} líneas en cada columna de la matriz densa, y son independientes en cada fila, afectando los accesos de cada fila a H líneas.

De esta forma, el número de fallos sobre la matriz producto puede estimarse como:

$$F_{\mathrm{A}} = \frac{MH}{L_{\mathrm{s}}}\left(1 + (L_{\mathrm{s}} - 1)S_{\mathrm{int\ np\ D}_0}\right) \tag{3.31}$$

3.5.3. Matriz `B`

Una vez más podemos modelar el comportamiento de esta matriz empleando un mecanismo similar al utilizado en el producto matriz dispersa-vector para el vector `X`. De hecho esta versión del algoritmo realiza el producto escalar de cada fila de la matriz dispersa por cada columna de la matriz `B`, siendo la diferencia con el orden JIK que en vez de multiplicar toda la matriz por cada columna de `B` antes de pasar a la siguiente, multiplica una fila de la matriz por todas las columnas de `B` antes de pasar a la siguiente fila.

El vector de área de interferencia asociado a las entradas que no son las primeras en generar un acceso a una línea dada durante un producto escalar ($N_{\mathrm{nz}} - pMN/L_{\mathrm{s}}$ en toda la matriz) es el mismo,

$$S_{\mathrm{int\ K\ B}} = S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(1) \tag{3.32}$$

En cuanto a los accesos generados por las primeras (pMN/L_{s}), calcularemos la probabilidad de acierto en el reuso de la línea en función del número de filas de la matriz dispersa procesadas desde el último acceso a la línea estudiada. Aquí el vector de área de interferencia cruzada será,

$$S_{\mathrm{cruzada\ B}}(i) = S_{\mathrm{s}}((i+1)\beta) \cup S_{\mathrm{s}}((i+1)\beta r) \cup S_{\mathrm{s}}(ir) \cup S_{\mathrm{r}}(H, i, M) \tag{3.33}$$

puesto que si se procesan i filas de la matriz dispersa entre ambos accesos, se leen los elementos de $i+1$ filas almacenados en los vectores `A` y `C` e i del vector `R`, además de calcularse los valores de i filas de la matriz producto. El motivo de que sean $i+1$ y no i las filas de la matriz dispersa cuyos datos se leen es que para el procesado de cada columna de la matriz `B` se leen todos los elementos de la fila procesada, con lo que los elementos de la fila en la

N	N_{nz}	p_{n}	H	C_{s}	L_{s}	K	σ	Δ
1	10	1.0 %	100	2	4	1	3.30	-0.18
1	10	1.0 %	100	4	4	4	0.18	0.01
1	10	1.0 %	100	8	4	1	0.20	0.10
1	10	1.0 %	100	8	8	2	0.27	-0.04
1	10	1.0 %	100	16	8	4	0.36	-0.14
1	10	1.0 %	100	64	8	2	1.05	-0.32
2	20	0.5 %	200	1	8	1	4.44	0.08
2	20	0.5 %	200	16	4	2	0.15	-0.00
2	20	0.5 %	200	32	8	4	0.18	-0.09
10	100	0.1 %	40	4	4	1	1.13	0.06
10	100	0.1 %	40	16	4	2	0.06	0.00
10	100	0.1 %	40	16	8	1	0.07	0.08
10	100	0.1 %	40	32	8	4	0.10	0.00
10	100	0.1 %	40	64	16	2	0.14	0.02
10	100	0.1 %	1000	256	32	2	0.05	0.35

Cuadro 3.5: Desviación del modelo para el producto matriz dispersa-matriz densa con orden IJK.

que se accede por última vez la línea de `B` estudiada serán leídos después del producto escalar en donde ésta es referenciada.

El vector de área de interferencia total $S_{\mathrm{int\ B}}(i)$ se calcula mediante (3.26), ya que el área accedida en `B` es la misma, variando únicamente que en vez de ser recorrida por filas para cada entrada de una fila de la matriz dispersa, se hace el producto escalar de todas las entradas de una fila por cada columna. También la expresión de $P_{\mathrm{ac\ B}}(j)$ y el promediado de estos valores para constituir $P_{\mathrm{ac\ B}}$ son los mismos que en el ordenamiento anterior. La expresión para el cálculo del número de fallos es también similar:

$$F_{\mathrm{B}} = H(pMN/L_{\mathrm{s}}(1 - P_{\mathrm{ac\ B}}) + (N_{\mathrm{nz}} - pMN/L_{\mathrm{s}})\, S_{\mathrm{int\ K\ B_0}}) \tag{3.34}$$

3.5.4. Validación y análisis

Incluimos en la tabla 3.5 los datos de validación correspondientes a las mismas combinaciones de los parámetros de entrada que se ilustraron en los dos ordenamientos previos posibles para los bucles de este algoritmo. En este caso resultó de estas pruebas un error medio para las predicciones del modelo de un 0.93 %, frente a una desviación típica media de las mediciones efectuadas en las simulaciones de un 3.26 %.

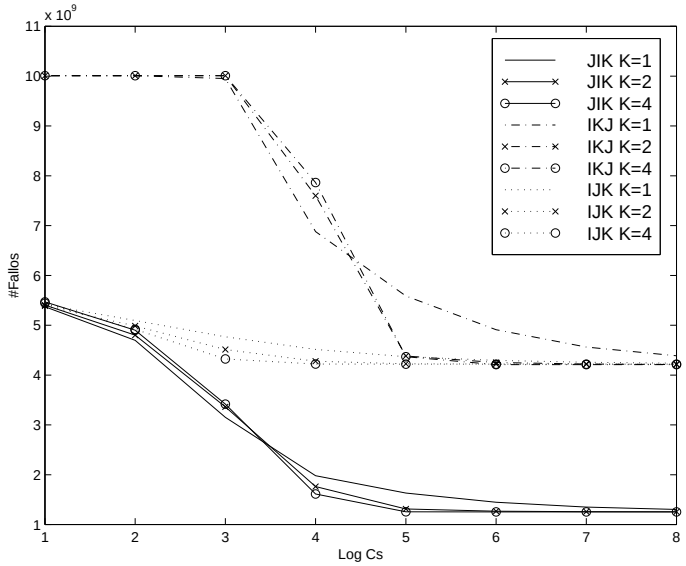

Figura 3.8: Número de fallos durante el producto matriz dispersa-matriz densa en función del ordenamiento, el tamaño de la caché y el grado de asociatividad para una matriz dispersa 10Kx10K con $p_{\mathrm{n}} = 0{,}05$ y $H = 1K$ usando $L_{\mathrm{s}} = 8$.

Realizamos a continuación un estudio comparativo del comportamiento de los algoritmos correspondientes a los tres ordenamientos.

Las figuras 3.8 y 3.9 muestran la evolución del número de fallos para los tres ordenamientos y tres niveles de asociatividad típicos en función del tamaño de la caché y de la línea. Vemos que el orden JIK es siempre el mejor para un tamaño de línea medio debido al gran predominio de los accesos secuenciales en éste frente a los otros dos ordenamientos. Sólo cuando el tamaño de la línea es muy reducido, no permitiendo, pues, explotar adecuadamente la localidad espacial, pueden los fallos de este ordenamiento ser superiores a los de los otros. De ellos, el IKJ es el menos eficiente, pues tanto la matriz densa `B` como la matriz producto `D` son accedidas únicamente por filas. El ordenamiento IJK permite al menos una explotación de la localidad espacial en el acceso a `B` durante el producto escalar de una columna de esta matriz con una fila de la matriz dispersa. También presenta una mayor localidad tanto espacial como temporal en el acceso a los elementos que conforman la matriz dispersa.

En lo tocante al tamaño de la caché, puede apreciarse como la mayor caída del número de fallos para el ordenamiento JIK en la figura 3.8 se da en la zona en la que C_{s} pasa de un valor muy inferior a N a uno mayor, dado

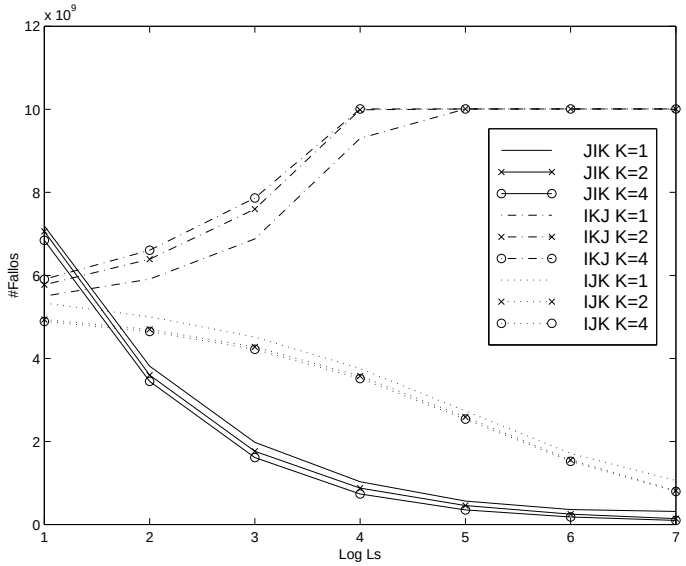

Figura 3.9: Número de fallos durante el producto matriz dispersa-matriz densa en función del ordenamiento, el tamaño de la línea de caché y el grado de asociatividad para una matriz dispersa 10Kx10K con $p_{\mathrm{n}} = 0{,}05$ y $H = 1K$ usando $C_{\mathrm{s}} = 16K$.

que las auto-interferencias en el acceso a la columna de la matriz `B` procesada en cada iteración del bucle en `J` desaparecen y las interferencias cruzadas se reducen fuertemente. El orden IKJ, por otra parte, es muy sensible a las auto-interferencias en el acceso a las líneas de una fila. En el caso de esta figura, $\mathrm{mcm}(N = 10K, C_{\mathrm{s}}) = 625C_{\mathrm{s}}$, con lo que las auto-interferencias surgen con los accesos a partir de la columna $C_{\mathrm{s}}/16$. Por ello, sólo para valores de $C_{\mathrm{s}} \geq 16K$ palabras podemos contener las 1000 líneas de una fila de la matriz sin que surjan estas interferencias y obtener aciertos en sus reusos. Dado que en el acceso a las líneas de `D` se tiene el mismo problema, el tamaño de 32K palabras es el primero en lograr una estabilización del número de fallos, influyendo en él decisivamente el grado de asociatividad mayor que uno para evitar que las líneas de `B` y `D` se expulsen mutuamente en sus conjuntos. Sin embargo, la gran probabilidad de interferencias cruzadas sigue haciendo que este orden se comporte peor que los otros dos. Para el ordenamiento IJK el tamaño de la caché influye muy poco porque `B` se accede por columnas y, como hemos dicho, los accesos a los componentes de la matriz dispersa tienen una distancia de reuso corta. El reuso de las líneas de `B` para distintas filas de la matriz dispersa es imposible para los tamaños de caché considerados, pues si tenemos un tamaño de línea de 8 palabras y una densidad 0.05, la probabilidad de acceso por línea de una columna de `B` es $p = 1 - (1 - 0{,}05)^8 = 0{,}336$, con lo

que durante el procesamiento de una fila de la matriz dispersa accederíamos por término medio a $0{,}336 \cdot (10000/8) \cdot 1000 \approx 420000$ líneas de `B`.

Es destacable también la reducida influencia en general del grado de asociatividad sobre la tasa de fallos: ésta aporta poco al orden JIK debido al predominio de los accesos secuenciales, los cuales eliminan la mayor parte de las interferencias. En el orden IJK la asociatividad puede reducir el número de fallos sobre los vectores `A` y `C`, pues la distancia entre reusos de sus datos no es grande, como hemos visto. Si el tamaño de la caché es lo suficientemente grande como para que estas interferencias cruzadas se minimicen, la asociatividad deja de influir en este ordenamiento, como puede verse en la figura 3.8. En el orden IKJ, hay una mayor dependencia de la relación entre el tamaño de la caché y de la línea para saber cuando ayuda la asociatividad debido a la mayor distancia entre los reusos. Por ejemplo, en el caso de la figura 3.9 una línea de `A`, `C`, `D` o `B` sólo se volvería a acceder tras referenciar unas $H = 1000$ líneas de `B` y otras tantas de `D`. Esto hace que en la caché de 16K palabras considerada los tamaños de línea mayores que $16000/2000 = 8$ reduzcan mucho de por sí la posibilidad de acierto en el reuso. Este efecto también puede apreciarse en la gráfica 3.8, en la que ya hemos explicado en el párrafo precedente por qué sólo a partir de un tamaño de caché de 16K palabras, y en especial de 32 Kpalabras, puede comenzar la asociatividad a reducir las tasas de fallos.

En cuanto a la influencia del tamaño de la línea, su incremento es beneficioso para los ordenamientos en función de su nivel de secuencialidad en los accesos, mejorando por tanto mucho el comportamiento del orden JIK y en menor medida el de IJK, y siendo contraproducente para el ordenamiento IKJ por su casi total carencia de posibilidades reales de explotación de la localidad espacial: en él, el aumento del tamaño de la línea sólo contribuye prácticamente a aumentar el tamaño del conjunto de trabajo donde se pueden dar reusos, como acabamos de explicar. Sólo valores muy elevados de L_s pueden dar lugar a crecimientos de la probabilidad de interferencia que eliminen las ventajas obtenidas del uso de la localidad espacial en las líneas grandes para JIK e IJK. Este efecto aumenta con la densidad de la matriz.

3.6. Trasposición de una matriz dispersa

En esta sección modelamos un algoritmo notablemente más complejo que los vistos hasta ahora: la trasposición de una matriz dispersa. Como puede verse en la figura 3.10, el código asociado, que ha sido extraído de [39],

```
1  DO I=2,N+1
     RT(I)=0
   END DO
2  DO I=1, R(M+1)-1
     J=C(I)+2
     RT(J)=RT(J)+1
   END DO
   RT(1)=1
   RT(2)=1
3  DO I=3, N+1
     RT(I)=RT(I)+RT(I-1)
   END DO
4  DO I=1, M
     DO K=R(I), R(I+1)-1
       J=C(K)
       P=RT(J)
       CT(P)=I
       AT(P)=A(K)
       RT(J)=P+1
     END DO
   END DO
```

Figura 3.10: Trasposición de una matriz dispersa.

presenta dos nuevos elementos con respecto a los códigos precedentes:

- Por un lado, este algoritmo no consta de un único grupo de bucles perfectamente anidados, sino que consta de una secuencia de cuatro anidamientos o bucles. Algunas estructuras de datos son referenciadas en varios (incluso en los cuatro) anidamientos, con lo que además de las probabilidades de acierto en el reuso de las porciones de las mismas accedidas previamente en un anidamiento dado, habremos de considerar la probabilidad de acierto al reusar un dato accedido en un bucle o anidamiento precedente.
- El número de niveles de indirección es notablemente mayor en el último anidamiento. Por ejemplo, puede apreciarse que `A(K)` se está copiando a `AT(RT(C(K)))`, siendo a su vez el índice `K` un valor comprendido entre `R(I)` y `R(I+1)-1`.

Como nuevas estructuras de datos tenemos los vectores `AT`, `CT` y `RT` que almacenarán la matriz traspuesta también en formato CRS.

3.6.1. Vector `A`

Este vector es accedido de una forma totalmente secuencial en el último bucle y, entre dos accesos consecutivos al mismo, se referencian cinco líneas pertenecientes a cuatro vectores diferentes. Además, cada β accesos aproximadamente hay una referencia más debido a la lectura de `R` al iniciar el

procesado de una nueva fila. Por tanto podemos estimar el número de fallos sobre el vector `A` como:

$$F_{\mathrm{A}} = \frac{N_{\mathrm{nz}}}{L_{\mathrm{s}}}\left(1 + (L_{\mathrm{s}} - 1)\frac{1}{\beta}\left((\beta - 1)\left(\bigcup^{5} S_{\mathrm{s}}(1)\right)_0 + \left(\bigcup^{6} S_{\mathrm{s}}(1)\right)_0\right)\right) \quad (3.35)$$

en donde denotamos como $\bigcup^{n} S$ la unión n veces de S consigo mismo y por $(\cdots)_0$ la componente 0 del vector de área de interferencia obtenido mediante la operación que aparece entre los paréntesis.

3.6.2. Vector `R`

Este vector sólo aparece en el último bucle y se accede de forma secuencial. Por tanto no presenta auto-interferencias. Las interferencias cruzadas vienen dadas por los accesos a los restantes vectores durante el procesado de una fila de la matriz dispersa original, o si se prefiere, durante una iteración del bucle más externo del anidamiento 4. Estos accesos afectan a β elementos consecutivos de los vectores `A` y `C` en la matriz original. En la matriz destino, el vector `RT` experimenta un acceso con probabilidad uniforme de referencia por línea, pues cada línea del mismo tiene una probabilidad $p_r = 1 - (1 - p_{\mathrm{n}})^{L_{\mathrm{s}}/r}$ de ser accedida, es decir, si hay al menos una entrada en el grupo de L_{s}/r columnas cuyas filas tendrían sus límites almacenados en una línea de `R`. El vector `AT` tiene un acceso por grupos de N_{nz}/N elementos (los correspondientes a una columna de la matriz original) con una probabilidad uniforme de referencia p_{n}. Este patrón de acceso es el modelado por S_{g}, descrito en la sección 2.3.3. Como siempre, el vector `CT` tiene un patrón idéntico, pero debiéndose tener en cuenta que está conformado por datos enteros. Así pues, el vector de área de interferencia correspondiente a `R` será:

$$\begin{aligned} S_{\mathrm{int\ R}} =& S_{\mathrm{s}}(\beta) \cup S_{\mathrm{s}}(\beta r) \cup S_{\mathrm{l}}(Nr, p_r) \cup \\ & S_{\mathrm{g}}(N_{\mathrm{nz}}, N_{\mathrm{nz}}/N, p_{\mathrm{n}}) \cup S_{\mathrm{g}}(N_{\mathrm{nz}} r, N_{\mathrm{nz}} r/N, p_{\mathrm{n}}) \end{aligned} \quad (3.36)$$

y el número de fallos se calculará como:

$$F_{\mathrm{R}} = \frac{Mr}{L_{\mathrm{s}}}\left(1 + (L_{\mathrm{s}}/r - 1) S_{\mathrm{int\ R}_0}\right) \quad (3.37)$$

3.6.3. Vectores `AT` y `CT`

Estos dos vectores siguen el mismo patrón de acceso, como puede verse en el cuarto anidamiento. Por tanto, al igual que se ha venido haciendo en los

algoritmos precedentes, explicaremos el modelado del comportamiento del vector `AT` y recordaremos que el del vector `CT` es idéntico, teniendo en cuenta la diferencia de escalado al estar constituido por elementos enteros en lugar de valores en punto flotante. El patrón de acceso es, como hemos mencionado, el modelado por S_{g}, con un valor para sus parámetros $n = N_{\mathrm{nz}}$, $t = N_{\mathrm{nz}}/N$ y $P_{\mathrm{g}} = p_{\mathrm{n}}$ para `AT`. Como suponemos que no hay porciones del vector en la caché al iniciarse la ejecución, el número de fallos puede estimarse como:

$$\begin{aligned} F_{\mathrm{AT}} = & \frac{N_{\mathrm{nz}}}{L_{\mathrm{s}}} + (N_{\mathrm{nz}} - N_{\mathrm{l\ AT}} M) S_{\mathrm{int\ K\ AT_0}} + \\ & \left(N_{\mathrm{l\ AT}} M - \frac{N_{\mathrm{nz}}}{L_{\mathrm{s}}} \right) (1 - P_{\mathrm{ac\ AT}}) \end{aligned} \tag{3.38}$$

ya que el primer acceso a cada línea será un fallo intrínseco y los restantes dependerán de la probabilidad de acierto en el reuso de la línea. Si $N_{\mathrm{l\ AT}}$ es el número medio de líneas de `AT` referenciadas durante el procesamiento de una fila de la matriz dispersa, calculado como:

$$N_{\mathrm{l\ AT}} = \begin{cases} N_{\mathrm{nz}}/L_{\mathrm{s}}(1 - (1 - p_{\mathrm{n}})^{L_{\mathrm{s}} N/N_{\mathrm{nz}}}) & \text{si } N_{\mathrm{nz}}/N \leq L_{\mathrm{s}} \\ \beta & \text{si } N_{\mathrm{nz}}/N > L_{\mathrm{s}} \end{cases} \tag{3.39}$$

habrá $N_{\mathrm{nz}} - N_{\mathrm{l\ AT}} M$ elementos que referenciarán una línea de `AT` accedida en la iteración previa del bucle en `K`, con lo que el vector de área de interferencia para ellos vendrá dado aproximadamente por los accesos a cuatro líneas diferentes, es decir:

$$S_{\mathrm{int\ K\ AT}} = \overset{4}{\bigcup} S_{\mathrm{s}}(1) \tag{3.40}$$

De los restantes $N_{\mathrm{l\ AT}} M$ accesos que son los primeros en referenciar una línea del vector `AT` durante el procesamiento de una fila de la matriz dispersa, $N_{\mathrm{nz}}/L_{\mathrm{s}}$ corresponden a los fallos intrínsecos y los restantes dependerán de la probabilidad de acierto en el reuso $P_{\mathrm{ac\ AT}}$. Este valor se calcula como:

$$P_{\mathrm{ac\ AT}} = \sum_{i=1}^{M-1} P_{\mathrm{ac}} (1 - P_{\mathrm{ac}})^{(i-1)} (1 - S_{\mathrm{int\ AT_0}}(i)) \tag{3.41}$$

donde cada sumando corresponde a la probabilidad de acierto en el reuso si desde el último acceso a la línea considerada se han procesado i filas de la matriz dispersa. Para derivarlo se emplea P_{ac}, la probabilidad de que la línea estudiada haya sido accedida en una iteración anterior del bucle en `I` del cuarto anidamiento:

$$P_{\mathrm{ac}} = 1 - (1 - p_{\mathrm{n}})^{\mathrm{máx}\{1, L_{\mathrm{s}} N/N_{\mathrm{nz}}\}} \tag{3.42}$$

El vector de interferencia durante el procesamiento de i filas se obtiene como:

$$S_{\text{int AT}}(i) = S^i_{\text{ga}}(N_{\text{nz}}, N_{\text{nz}}/N, p_{\text{n}}) \cup S_{\text{cruzada AT}}\ (i) \tag{3.43}$$

en donde la obtención del vector de área de auto-interferencia de un acceso por grupos ya se explicó al final del apartado 2.3.3, siendo ésta una aplicación concreta, y el vector de área de interferencia cruzada viene dado por:

$$\begin{aligned} S_{\text{cruzada AT}}(i) =& S_{\text{s}}(\beta i) \cup S_{\text{s}}(\beta ir) \cup S_{\text{s}}(ir) \cup \\ & S^i_{\text{l}}(Nr, p_r) \cup S^i_{\text{g}}(N_{\text{nz}}r, N_{\text{nz}}r/N, p_{\text{n}}) \end{aligned} \tag{3.44}$$

puesto que durante la trasposición de i filas de la matriz dispersa se leen β componentes en secuencia por cada fila en los vectores `A` y `C` y uno por fila en `R`. Por otra parte, el vector `RT` tiene un acceso con probabilidad uniforme de referencia por línea, como se explicó en el apartado anterior, y `CT` ya hemos dicho que tiene un acceso por grupos análogo al de `AT` y que también se ilustró en el apartado 3.6.2.

3.6.4. Vector `C`

Este vector es el único junto con `RT` que se referencia en varios bucles del algoritmo. Debemos por tanto considerar la posibilidad de que porciones de estos vectores estén presentes en la caché al inicio de la ejecución de un nuevo bucle, aspecto que no había aparecido en ninguno de los algoritmos anteriores. En particular, el vector `C` participa en los bucles de dos anidamientos del algoritmo (dos y cuatro en la figura 3.10). Calcularemos por separado el número de fallos que generan las referencias a esta estructura en ambos bucles. En general, representaremos como F_{V_i} el número de fallos sobre la estructura de datos `V` en el anidamiento i cuando ésta pueda aparecer en varios anidamientos.

Bucle 2

En el bucle dos este vector presenta un acceso puramente secuencial en el que es fácil ver que el número de fallos según nuestro modelo se puede computar como:

$$F_{\text{C}_2} = \frac{N_{\text{nz}}r}{L_{\text{s}}}\left(1 + (L_{\text{s}}/r - 1)S_{\text{s}_0}(1)\right) \tag{3.45}$$

al haber sólo un acceso a otra línea entre dos accesos consecutivos a una línea de `C`.

Bucle 4

En el cuarto anidamiento este vector vuelve a presentar un acceso secuencial en el que podemos apreciar que entre dos accesos sucesivos a la misma línea se accede a cuatro líneas de otros tantos vectores, exceptuando el caso en el que se finaliza el procesamiento de una fila, en el que se accede además a una línea del vector `R`. El vector de interferencia correspondiente puede calcularse usando una media, de la misma forma que se hizo en el apartado 3.6.1:

$$S_{\text{int C}_4} = \frac{1}{\beta}\left((\beta-1)\left(\bigcup^{4} S_{\text{s}}(1)\right)_0 + \left(\bigcup^{5} S_{\text{s}}(1)\right)_0\right) \tag{3.46}$$

Dado que este vector ha sido accedido previamente, no puede asumirse que el primer acceso a cada línea vaya a resultar en un fallo, con lo que la expresión del número de fallos sobre el vector `R` en este último anidamiento será:

$$F_{\text{C}_4} = \frac{N_{\text{nz}} r}{L_{\text{s}}}\left(S_{\text{int ext C}_{4_0}} + (L_{\text{s}}/r - 1) S_{\text{int C}_{4_0}}\right) \tag{3.47}$$

donde $S_{\text{int ext C}_4}$ será el vector de área de interferencia medio para las líneas que componen el vector `C` generados por los accesos comprendidos entre la referencia a cada línea en el bucle dos y el nuevo acceso en el cuarto anidamiento. Para calcularlo indexamos dichas líneas de 0 a $\lceil N_{\text{nz}} r/L_{\text{s}} \rceil - 1$ y calculamos $S_{\text{int ext C}_4}(i)$, el vector de área de interferencia para la i-ésima línea como:

$$\begin{aligned} S_{\text{int ext C}_4}(i) = & S_{\text{auto C}_4}(i) \cup S_{\text{s}}(Nr) \cup S_{\text{AT}}(i) \cup \\ & S_{\text{CT}}(i) \cup S_{\text{R}}(i) \cup S_{\text{A}}(i) \end{aligned} \tag{3.48}$$

en donde si $l = N_{\text{nz}} r / C_{\text{sk}}$ el vector de área de auto-interferencia se calcula como:

$$S_{\text{auto C}_4}(i) \begin{cases} S_{\text{s}}(\lceil l \rceil C_{\text{sk}}) & \text{si } (l - \lfloor l \rfloor) N_{\text{k}} > i \text{ mód } N_{\text{k}} \\ S_{\text{s}}(\lfloor l \rfloor C_{\text{sk}}) & \text{en caso contrario} \end{cases} \tag{3.49}$$

ya que si $(l - \lfloor l \rfloor) N_{\text{k}} > i$ mód N_{k}, la i-ésima línea del vector `C` compite en su conjunto con otras $\lceil l \rceil$ líneas de dicho vector. En caso contrario, hay $\lfloor l \rfloor$ líneas que puedas provocarle auto-interferencias.

Por otro lado, el vector `RT` es accedido en su totalidad en el bucle 3, con lo que podemos estimar directamente su vector de área de interferencia cruzada como $S_{\text{s}}(Nr)$, al constar de N elementos enteros. Para calcular el vector de áre de interferencia cruzada para los restantes vectores, accedidos sólo en el cuarto anidamiento, tendremos en cuenta que al llegar el momento

de procesar la i-ésima línea de `C` se habrán tratado ya los iL_s/r elementos de las líneas precedentes, que corresponden aproximadamente a $iL_s/(\beta r)$ filas de la matriz dispersa. Con este dato tendremos que:

$$S_{\mathrm{AT}}(i) = S_{\mathrm{g}}^{(iL_s)/(\beta r)}(N_{\mathrm{nz}}, N_{\mathrm{nz}}/N, p_{\mathrm{n}}) \quad (3.50)$$

$$S_{\mathrm{CT}}(i) = S_{\mathrm{g}}^{(iL_s)/(\beta r)}(N_{\mathrm{nz}}r, N_{\mathrm{nz}}r/N, p_{\mathrm{n}}) \quad (3.51)$$

$$S_{\mathrm{R}}(i) = S_{\mathrm{s}}(i(L_s/\beta)) \quad (3.52)$$

$$S_{\mathrm{A}}(i) = S_{\mathrm{s}}(i(L_s/r)) \quad (3.53)$$

3.6.5. Vector `RT`

Este vector es referenciado en los cuatro anidamientos del algoritmo. En el primero tiene un acceso puramente secuencial en el que se generan Nr/L_s fallos intrínsecos, no habiendo vector de área de interferencia alguno.

Bucle 2

En el segundo bucle los accesos a `RT` siguen un patrón idéntico al vector `X` en el producto matriz dispersa-vector (direccionamiento a través de los valores del vector `C` leyendo éste secuencialmente). Las diferencias radican en que sólo hay una posible fuente de interferencias cruzadas (el acceso al vector `C` y que el vector `RT` está constituido por enteros en lugar de por valores en punto flotante; además ahora hay una probabilidad de acierto en el primer acceso a una línea ya que se trata de un reuso. De esta forma, el número de fallos en este segundo bucle se calculará como:

$$F_{\mathrm{RT_2}} = pM\frac{Nr}{L_s}(1 - P_{\mathrm{ac\ RT_2}} - P_{\mathrm{ac\ ext\ RT_2}}) + \left(N_{\mathrm{nz}} - pM\frac{Nr}{L_s}\right) S_{\mathrm{s_0}}(1) \quad (3.54)$$

puesto que entre dos accesos a la misma línea durante el procesado de las entradas de una fila sólo hay un acceso a una línea del vector `C`. La probabilidad de acierto en los accesos que no son los primeros a una línea, $P_{\mathrm{ac\ RT_2}}$ se calcula de forma análoga a la expuesta en (3.10) y (3.11) teniendo en cuenta que en lugar de p debe usarse p_r, introducido en la sección 3.6.2, y que el valor de $S_{\mathrm{int\ RT_2}}(i)$ será:

$$S_{\mathrm{int\ RT_2}}(i) = S_{\mathrm{la}}^{i}(Nr, p_r) \cup S_{\mathrm{s}}(i\beta r) \quad (3.55)$$

al constar `RT` de N elementos enteros y accederse $i\beta$ elementos del vector `C` durante el procesamiento correspondiente a i filas de la matriz dispersa.

Para calcular $P_{\text{ac ext RT}_2}$, la probabilidad de acierto en el primer acceso a cada línea durante este segundo bucle, numeraremos las $N_{\text{l RT}} = \lceil Nr/L_{\text{s}} \rceil$ líneas de `RT` de 0 a $N_{\text{l RT}} - 1$, de forma similar a como se hizo en el apartado anterior. Este valor será la media aritmética de las probabilidades $P_{\text{ac ext RT}_2}(i)$ correspondientes a cada línea, que se calculan como:

$$P_{\text{ac ext RT}_2}(i) = \frac{1}{M} \sum_{j=1}^{M} (1 - p_r)^{(j-1)} (1 - S_{\text{int ext RT}_{2_0}}(i, j)) \tag{3.56}$$

en donde cada término del sumatorio representa la probabilidad de que la línea no haya sido accedida antes del procesamiento de la fila j-ésima en el segundo bucle multiplicada por la probabilidad de que haya sobrevivido en la caché desde su acceso previo en el primer bucle. Este valor viene dado por la expresión:

$$\begin{aligned} S_{\text{int ext RT}_2}(i, j) = & S_{\text{s}}(\lfloor (N_{\text{l RT}} - 1 - i)/N_{\text{k}} \rfloor C_{\text{sk}}) \cup \\ & S_{\text{l}}(\lfloor i/N_{\text{k}} \rfloor C_{\text{sk}}, 1 - (1 - p_r)^j) \cup S_{\text{s}}((j - 1/2)\beta r) \end{aligned} \tag{3.57}$$

en donde el primer componente representa el vector de interferencia generado por las líneas del vector `RT` que se accedieron en el primer bucle después de la i-ésima y el segundo da el vector de área de auto-interferencia que las i líneas precedentes generan a lo largo del procesamiento de j filas de la matriz dispersa. Por último, se añade el vector de área de interferencia cruzada asociado a los accesos a los componentes del vector `C` durante el procesamiento de las $j - 1$ filas precedentes y aproximadamente la mitad de la actual.

Bucle 3

Al pasar a considerar el tercer bucle, se observa que se trata de un acceso puramente secuencial en que sólo pueden darse fallos en los accesos que sean los primeros a una línea. La probabilidad media de acierto en el reuso de dichas líneas es exactamente $P_{\text{ac ext RT}_2}$, calculada para el bucle precedente. El motivo es la simetría del comportamiento del segundo bucle con respecto al primero y del tercero respecto al segundo: ahora la línea i-ésima tiene accesos secuenciales a i líneas antes que ella en este tercer bucle y hay $N_{\text{l RT}} - 1 - i$ líneas que pueden haberle provocado auto-interferencias si no se ha accedido durante el procesamiento de las j últimas filas. Y el comportamiento de las interferencias cruzadas es idéntico. La simetría del comportamiento de las línea del vector `RT` en el paso de un bucle al siguiente en estos dos casos hace que los valores de $S_{\text{int ext RT}}(i, j)$ sean simétricos respecto a i. Como

consecuencia la media de estos valores es la misma. Por tanto el número de fallos en este bucle se estima como:

$$F_{\mathrm{RT3}} = Nr/L_{\mathrm{s}}(1 - P_{\mathrm{ac\ ext\ RT_2}}) \tag{3.58}$$

Bucle 4

Finalmente, el cuarto bucle tiene un comportamiento muy similar al segundo, puesto que la estructura del patrón de referencias al vector `RT` es el mismo, e incluso también se parte de un acceso previo secuencial al vector. Así pues, el número de fallos puede estimarse usando la expresión (3.54), si bien se requieren algunas modificaciones en el cálculo de sus parámetros. Así, para obtener $S_{\mathrm{int\ RT_4}}(i)$, del que se deriva $P_{\mathrm{ac\ RT_4}}$, habremos de tener en cuenta que en este bucle tenemos como vector de área de interferencia cruzada:

$$\begin{aligned} S_{\mathrm{cruzada\ RT_4}}(i) =& S_{\mathrm{s}}(i\beta) \cup S_{\mathrm{s}}(i\beta r) \cup S_{\mathrm{s}}(ir) \cup \\ & S^i_{\mathrm{g}}(N_{\mathrm{nz}}, N_{\mathrm{nz}}/N, p_{\mathrm{n}}) \cup S^i_{\mathrm{g}}(N_{\mathrm{nz}}r, N_{\mathrm{nz}}r/N, p_{\mathrm{n}}) \end{aligned} \tag{3.59}$$

en lugar de $S_{\mathrm{s}}(i\beta r)$ en (3.55). Su composición es, como cabría esperar, la de $S_{\mathrm{cruzada\ AT}}(i)$ en (3.44) reemplazando el vector de área de interferencia cruzada asociado a `RT`, que correspondía a un acceso con probabilidad uniforme de referencia por línea, por el asociado a `AT`, con su acceso por grupos de elementos.

Por el mismo motivo, para obtener $S_{\mathrm{int\ ext\ RT_4}}(i,j)$ a partir de la expresión (3.57), se reemplaza el cálculo de la interferencia cruzada $S_{\mathrm{s}}((j-1/2)\beta r)$ por $S_{\mathrm{cruzada\ RT_4}}(j-1/2)$, siendo idéntico el resto del proceso de cálculo de $P_{\mathrm{ac\ ext\ RT_4}}$.

Por último, la precisión puede aumentarse teniendo en cuenta la probabilidad de que se de un fallo entre los dos accesos al mismo elemento de `RT` que se dan en el bucle interno del anidamiento. Entre ambos se accede a tres líneas distintas, con lo que el vector de área de interferencia cruzada es $S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(1)$; y el número de accesos es N_{nz}, uno por entrada. Sin embargo, en nuestro modelado hemos supuesto un compilador con cierta capacidad de optimización para los accesos que ejecuta la escritura del valor en `RT` inmediatamente después de su lectura, evitando toda posibilidad de interferencia cruzada.

N	N_{nz}	p_{n}	C_{s}	L_{s}	K	σ	Δ
1	10	1.0 %	2	4	1	0.70	0.01
1	10	1.0 %	4	4	4	0.24	-0.31
1	10	1.0 %	8	4	1	2.02	-1.91
1	10	1.0 %	8	8	2	1.25	1.27
1	10	1.0 %	16	8	4	0.58	2.03
1	10	1.0 %	64	8	2	13.05	-3.96
1	100	10.0 %	1	8	1	0.1	0.28
1	100	10.0 %	16	4	2	0.4	-0.59
1	100	10.0 %	32	8	2	1.7	-1.84
10	100	0.1 %	8	8	2	0.08	0.14
10	100	0.1 %	32	8	2	0.20	0.84
10	100	0.1 %	64	16	1	1.44	0.85
10	100	0.1 %	64	8	4	0.26	1.97
10	100	0.1 %	128	16	2	1.89	1.69
10	100	0.1 %	256	32	2	16.20	-1.46

Cuadro 3.6: Desviación del modelo para la trasposición de una matriz dispersa.

3.6.6. Validación y análisis

En el caso de este algoritmo, se ha obtenido un error medio en la predicción de un 1.61 %, y alcanzando la desviación típica media de los valores de fallos medidos el 8.91 % de su media. La tabla 3.6 muestra estos valores para algunas combinaciones concretas de los parámetros del algoritmo.

El comportamiento del algoritmo en los casos análogos a los estudiados en las figuras 3.2 y 3.3 es idéntico. No obstante, el mayor tamaño del conjunto de trabajo de este algoritmo precisa de tamaños de caché mayores para observar muchos efectos. Por ejemplo, la figura 3.11 muestra la trasposición de la matriz tratada en la figura 3.3 en una caché de 256K palabras. Podemos observar que aquí el mínimo de fallos se ha dado para un tamaño de línea de unas 8 palabras. La razón es que teniendo una densidad 0.18, durante el procesamiento de cada fila de la matriz dispersa se acceden aproximadamente $10000 \cdot 0{,}18 = 1800$ líneas diferentes en los vectores `AT` y `CT`, que son los que tienen el mayor conjunto de trabajo. Por otra parte, con esta densidad, habrá por término medio una entrada por cada columna, es decir, un nuevo acceso a una línea previamente accedida de los vectores `AT` y `CT`, cada $1/0{,}18 = 5{,}55$ filas de la matriz dispersa. Esto nos da un conjunto de $1800 \cdot 5{,}55 \cdot 2 \approx 20000$ líneas como conjunto de acceso en ambos vectores. Si

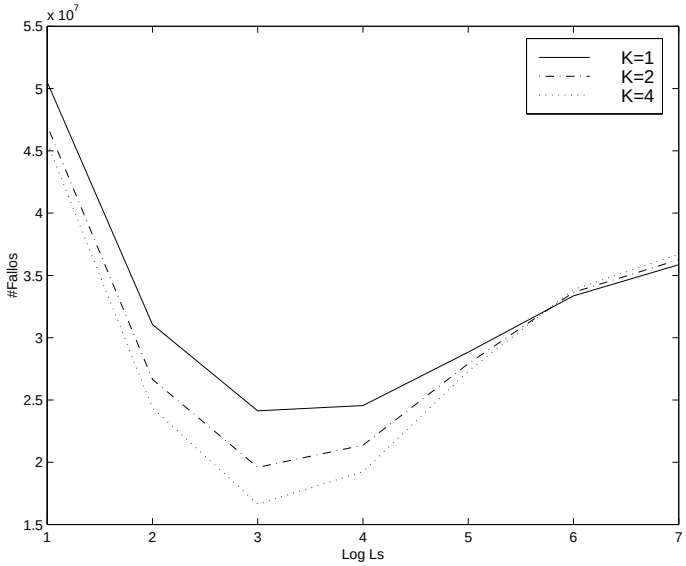

Figura 3.11: Número de fallos durante la trasposición de una matriz $10K \times 10K$ con $p_n = 0{,}18$ en una caché de 256K palabras en función de L_s para varios grados de asociatividad.

dividimos 256K palabras entre 20000 líneas obtenemos un tamaño por línea aproximado de 13 palabras. Dado que el conjunto de trabajo debe incluir también las líneas de los otros vectores, y el número de líneas requerido para mantenerlo debe ser mayor o igual a 20000, el valor óptimo de la línea es la potencia de dos inmediatamente inferior: ocho.

Respecto a las asociatividades, cuando tenemos un gran número de líneas, ayudan a reducir el efecto de las interferencias; pero cuando éste se reduce, el elevado número de accesos que se dan a cada conjunto al no poder contener la caché el conjunto de trabajo, hace que los niveles elevados de asociatividad se comporten peor al hacer que las interferencias afecten a la totalidad de las líneas del conjunto.

La figura 3.12 muestra la distribución entre los vectores empleados en el algoritmo de los fallos presentados en la gráfica 3.11 para $K = 2$. Con ella se demuestra que para los tamaños de línea superiores a 8 la gran mayoría de los fallos están asociados a los vectores `AT` y `CT`, que tienen el mismo comportamiento. Con los tamaños inferiores se logra mantener el conjunto de trabajo de dichos vectores, pero desperdiciamos la localidad espacial presente en los accesos a los vectores `C` y `A`.

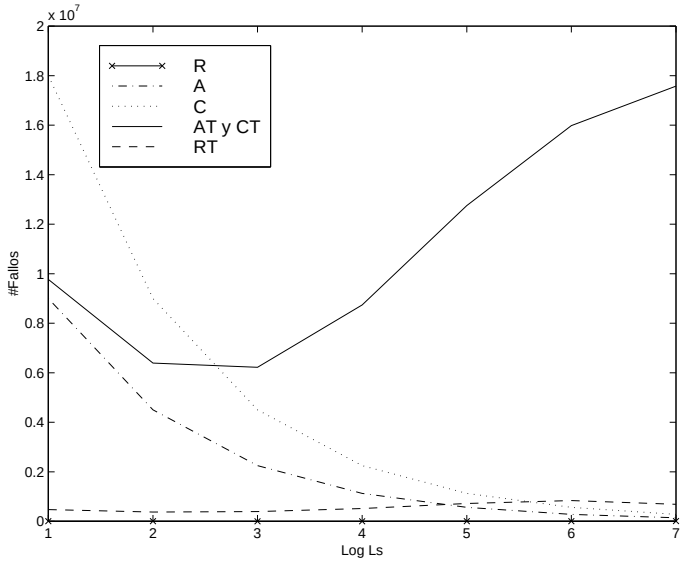

Figura 3.12: Número de fallos desglosado por vectores durante la trasposición de una matriz $10K \times 10K$ con $p_n = 0{,}18$ en una caché de dos vías y 256K palabras en función de L_s.

3.7. Producto matriz dispersa-matriz densa con orden IKJ altamente optimizado

Como último algoritmo empleado para ilustrar la aplicación de nuestro modelo hemos escogido una versión del producto matriz dispersa-matriz densa con orden IKJ introducido en la sección 3.4 que ha sido optimizada para su ejecución en una máquina con un moderno procesador superescalar. El modelado del algoritmo ha sido introducido en [22] para cachés de mapeado directo, y en [21] para cachés de una asociatividad arbitraria. El código para este algoritmo, propuesto en [38], se muestra en la figura 3.13, en donde se pueden apreciar las siguientes mejoras:

- Presenta un *blocking* a nivel de caché, procesando un bloque de B_{K} filas por B_{J} columnas de la matriz densa `B`. Dado que el *blocking* sobre la dimensión K afecta a la matriz dispersa, se emplea un vector auxiliar `R2` que indica la posición de la primera entrada en cada fila que es susceptible de pertenecer al bloque que se está procesando.

- En el orden IKJ el acceso a la matriz densa en el bucle más interno es por filas. El algoritmo realiza una copia del bloque a una matriz

```
 1 DO J2=1, H, BJ
 2   LIMJ=J2+MIN(BJ, H-J2+1)-1

 3   DO I=1, M+1
 4     R2(I)=R(I)
 5   ENDDO

 6   DO K2=1, N, BK
 7     LIMK=K2+MIN(BK, N-K2+1)

 8     DO J=1, LIMJ-J2+1
 9       DO K=1, LIMK-K2
10         WB(J,K)=B(K2+K-1,J2+J-1)
11       ENDDO
12     ENDDO

13     DO I=1, M
14       K=R2(I)
15       LK=R(I+1)

16       d1=D(I,J2)

17       d2=D(I,J2+1)
         ...
18       dbj=D(I,J2+BJ-1)
                  ↪
19       DO WHILE (K<LK AND C(K)<LIMK)
20         a=A(K)
21         ind=C(K)
22         d1=d1+a*WB(1,ind-j2+1)
23         d2=d2+a*WB(2,ind-j2+1)
           ...
24         dbj=dbj+a*WB(BJ, ind-j2+1)
25         K=K+1
26       ENDDO

27       D(I,J2)=d1
28       D(I,J2+1)=d2
         ...
29       D(I,J2+BJ-1)=dbj

30       R2(I)=K
31     ENDDO
32   ENDDO
33 ENDDO
```

Figura 3.13: Producto matriz dispersa-matriz densa con orden IKJ y *blocking* en los niveles de memoria y registros.

de almacenamiento temporal `WB` efectuando una trasposición (líneas 8–12). De esta forma estos accesos pasan a ser secuenciales, con lo que se evitan las posibles auto-interferencias y se mejora la capacidad de explotación de la localidad espacial.

- Hay también un *blocking* bidimensional a nivel de registros. Esta transformación del código comienza con la aplicación de strip mining al bucle más interno de un algoritmo sin *blocking*, obteniéndose un nuevo bucle interno de computación y un bucle de control, y después intercambiando este último con alguno de los bucles que lo comprenden. En este caso, al estar ya afectado por el *blocking* el bucle más interno, no es preciso pasar por esta fase. Posteriormente se desenrolla por completo el bucle interno y se extraen del bucle que lo comprende las operaciones de carga y almacenamiento en memoria que afectan a posiciones que pueden ser reusadas. Esta técnica limita, pues, el tamaño de este bucle interno en función del número de registros disponibles para almacenar los valores de estas posiciones. Sin embargo, da lugar a un número menor de accesos a memoria por operación aritmética e incremente el número de operaciones numéricas independientes en el cuerpo del bucle. En nuestro caso, el bucle interno controlado por la variable `J` ha

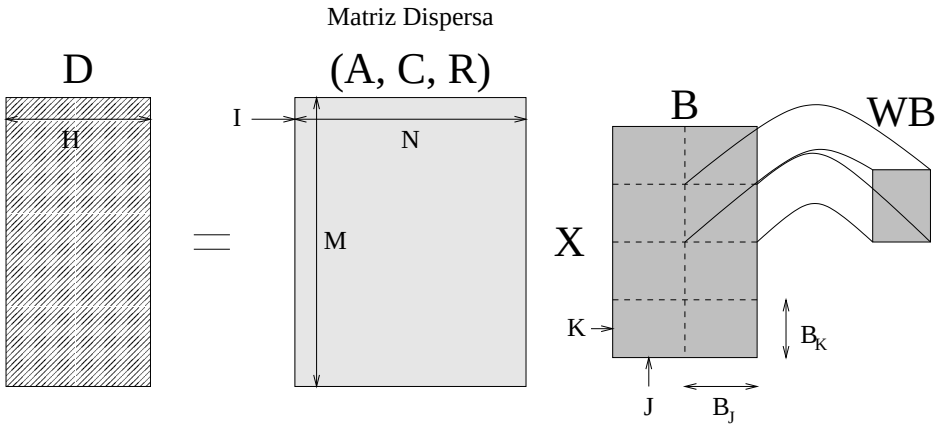

Figura 3.14: Representación gráfica de las matrices implicadas en el producto matriz dispersa-matriz densa con orden IKJ optimizado y su relación.

sido el afectado por este cambio, extrayéndose del bucle en `K` las cargas y almacenamiento de los valores de la matriz producto `D`. Esta modificación requiere el uso de B_J registros (`d1`, `d2`, ..., `dbj` en la figura) para almacenar dichos valores.

La figura 3.14 presenta la relación gráfica entre las matrices implicadas en la ejecución del algoritmo, incluyendo sus tamaños y los índices de los bucles que rigen cada dimensión.

En los siguientes apartados se modela el comportamiento de cada estructura de datos referenciada en el algoritmo. En este código surgen nuevos parámetros a los que haremos referencia en las ecuaciones: los ya mencionados B_J y B_K, que nos dan el tamaño del bloque en las dimensiones J y K, y el número correspondiente de bloques en cada dimensión, $N_{B_J} = H/B_J$ y $N_{B_K} = N/B_K$. A fin de simplificar el modelado asumiremos que B_J es divisor de H y B_K es divisor de N. En la práctica se ha comprobado que si el tamaño del bloque de una dimensión no es divisor del tamaño de la misma, la estimación es buena redondeando hacia arriba el número de bloques resultantes ($N_{B_J} = \lceil H/B_J \rceil$ y $N_{B_K} = \lceil N/B_K \rceil$, respectivamente) si el número de bloques en la dimensión no es muy pequeño.

3.7.1. Vector `R`

El patrón de accesos en `R` y `R2` son muy similares, por lo que consideraremos primero los accesos a uno de ellos (`R`, por ejemplo) y supondremos

que el número de fallos cometidos accediendo al otro es el mismo, lo cual ha probado ser una buena aproximación. En el siguiente apartado describiremos las diferencias para el modelado del vector `R2`.

Podemos descomponer los accesos a `R` en dos grandes grupos: los que se realizan durante el copiado de `R` a `R2` y los que se efectúan en el bucle I (ver bucles en líneas 3 y 13 del código en la figura 3.13, respectivamente). Calcularemos en primer lugar los primeros.

Fallos en la copia de `R` a `R2`

La primera vez tendremos un fallo intrínseco cada L_s/r accesos, pero las N_{B_J} restantes habrá una probabilidad de acierto en el reuso diferente para cada una de las $N_{lR} = Nr/L_s$ líneas del vector `R`. Considerando el bucle en `I` de la línea 13, que es en donde se lee este vector entre dos copias sucesivas, podemos calcular el vector de área de interferencia para la i-ésima línea como:

$$\begin{aligned} S_{\text{int ext Rc}}(i) =& S_s(C(Nr)C_{sk}) \cup S_s(Nr) \cup S_r(B_J, iL_s/r, M) \cup \\ & S_l^{iL_s/r}(B_J B_K, p_{WB}) \cup S_l(\beta i L_s/r, p_A) \cup S_l(\beta i/r L_s r, p_C) \end{aligned} \tag{3.60}$$

donde el primer término representa el vector de área de auto-interferencia del vector y los demás corresponden, en este orden, a los vectores de área de interferencia cruzada asociados a:

- `R2`, puesto que desde que se accede a la línea i-ésima de `R` en el bucle hasta que se la vuelve a acceder en el copiado, se accede a la totalidad de `R2`.
- `D`, ya que por cada línea procesada de `R` se leen L_s/r filas de `D` correspondientes a un bloque de B_J columnas, con una distancia entre regiones M, el número de filas.
- `WB`, que tiene una probabilidad uniforme de acceso por línea en cada iteración del bucle en `I` que estudiamos que puede ser estimada como:

  $$p_{WB} == 1 - (1 - p_n)^{\text{Grupos}(B_J, L_s)} \tag{3.61}$$

  en donde la función Grupos(a, b) proporciona el número medio de grupos diferentes de a elementos consecutivos en los que se puede dividir un vector infinito abstracto que recaen en un grupo de b elementos consecutivos de dicho vector. Su fórmula viene dada por:

  $$\text{Grupos}(a, b) = \begin{cases} 1 & \text{si } a \text{ mód } b = 0 \\ \frac{b + a - \text{mcd}(b, a \text{ mód } b)}{a} & \text{en otro caso} \end{cases} \tag{3.62}$$

de forma que Grupos$(B_\mathrm{J}, L_\mathrm{s})$ da la media de columnas distintas de `WB` por línea de caché de esta matriz. Dado que se accede una columna si y sólo hay una entrada en la columna correspondiente de la fila del bloque de la matriz dispersa que se está procesando, la derivación de (3.61) es directa. Por otra parte, por cada línea procesada de `R` se realizan L_s/r iteraciones del bucle, con lo que obtenemos la expresión $S_\mathrm{l}^{iL_\mathrm{s}/r}(B_\mathrm{J}B_\mathrm{K}, p_\mathrm{WB})$.

- `A`, teniendo en cuenta que este vector consta de β componentes por fila de la matriz dispersa. La probabilidad de acceso a una línea dada de la región de $\beta L_\mathrm{s}/r$ elementos correspondiente al procesamiento de irL_s filas de la matriz dispersa puede estimarse como:

$$p_\mathrm{A} = \frac{B_\mathrm{K}p_\mathrm{n} + (L_\mathrm{s}-1)(1-p_{\text{comp A}})(1-(1-p_\mathrm{n})^{B_\mathrm{K}})}{\beta} \tag{3.63}$$

es decir, la proporción que sobre los β elementos de una fila representan los $B_\mathrm{K}p_\mathrm{n}$ elementos accedidos por término medio al procesar una submatriz más los $L_\mathrm{s}-1$ elementos adicionales que se traen a la caché al efectuar un acceso secuencial. Sin embargo, en este caso este último valor se multiplica por la probabilidad de que las regiones de `A` accedidas durante el procesamiento de dos filas de un bloque consecutivas no compartan una línea, pues en ese caso los elementos adicionales son trasladados una sola vez a la caché para ambos grupos, y no dos. La probabilidad $p_{\text{comp A}}$ de que la compartan se computa como:

$$p_{\text{comp A}} = \frac{1}{L_\mathrm{s}} \sum_{j=0}^{L_\mathrm{s}-1} \binom{N-B_\mathrm{K}}{j} p_\mathrm{n}^j (1-p_\mathrm{n})^{(N-B_\mathrm{K}-j)} (L_\mathrm{s}-1-j) \tag{3.64}$$

en donde en cada término del sumatorio se multiplica la probabilidad de que en las $N-B_\mathrm{K}$ posiciones de la matriz que separan la última posición accesible de una fila por la primera de la siguiente, accediendo por filas, haya exactamente j entradas por el número de posiciones posibles en una línea en las que este hecho daría lugar a que el siguiente grupo de elementos a procesar se iniciase en ella.

Debemos hacer notar que en caso de matrices muy dispersas, puede haber filas en las que no se acceda a ningún elemento durante el procesamiento de un bloque. En estos casos, $p_{\text{comp A}}$ suele tener un valor alto, y el sumar $L_\mathrm{s}-1$ al número medio de elementos accedidos por fila del bloque, que es muy reducido, es erróneo. De ahí que se multiplique este valor también por $1-(1-p_\mathrm{n})^{B_\mathrm{K}}$, la probabilidad de que una fila del bloque no esté vacía.

- Por último tenemos el vector `C`, con un comportamiento muy similar al anterior. En este caso calcularemos p_{C} como:

$$p_{\mathrm{C}} = \frac{(B_{\mathrm{K}} p_{\mathrm{n}} + \mathrm{Menor}(B_{\mathrm{K}}, N))r + (L_{\mathrm{s}} - 1)(1 - p_{\mathrm{comp\ C}})}{\beta r} \tag{3.65}$$

  al estar compuesto por enteros y porque, como puede verse en la línea 19 del algoritmo, el vector `C` es accedido por cada fila del bloque, independientemente de que esté vacía o no. En el caso de que haya más de un bloque ($B_{\mathrm{K}} < N$) se accede a un elemento adicional para detectar el final de la fila del bloque considerado (véase la definición de $\mathrm{Menor}(a, b)$ en (2.26)). La probabilidad de compartición de una línea por los grupos de elementos correspondientes al procesamiento de dos filas consecutivas del bloque es:

$$p_{\mathrm{comp\ C}} = \frac{r}{L_{\mathrm{s}}} \sum_{j=0}^{\frac{L_{\mathrm{s}}}{r}-1} \binom{N - B_{\mathrm{K}}}{j} p_{\mathrm{n}}^{j} (1 - p_{\mathrm{n}})^{(N - B_{\mathrm{K}} - j)} (L_{\mathrm{s}}/r - 1 - j) \tag{3.66}$$

  radicando la diferencia con el valor correspondiente del vector anterior en el posible distinto tamaño de sus componentes.

De esta forma, el número de fallos sobre `R` en la copia puede estimarse como:

$$\begin{aligned} F_{\mathrm{copia\ R}} =& N_{\mathrm{l\ R}} + (N_{B_{\mathrm{J}}} - 1) \sum_{i=1}^{N_{\mathrm{l\ R}}} S_{\mathrm{int\ ext\ Rc_0}}(i) + \\ & N_{B_{\mathrm{J}}} (Nr - N_{\mathrm{l\ R}}) S_{\mathrm{s_0}}(1) \end{aligned} \tag{3.67}$$

en donde el último sumando corresponde a la posibilidad de que el acceso a `R2` entre dos accesos consecutivos y secuenciales a una línea de `R` en el copiado genere interferencias.

Fallos en el bucle principal en `I`

De forma similar, calcularemos el vector de interferencia para cada línea de `R` desde el proceso de copia previamente modelado:

$$\begin{aligned} S_{\mathrm{int\ ext\ c\ RI}}(i) =& S_{\mathrm{s}}(C(Nr)C_{\mathrm{sk}}) \cup S_{\mathrm{s}}(Nr) \cup S_{\mathrm{r}}(N_{B_{\mathrm{J}}}, B_{\mathrm{K}}, N) \cup \\ & S_{\mathrm{s}}(B_{\mathrm{J}} B_{\mathrm{K}}) \cup S_{\mathrm{l}}(\beta i L_{\mathrm{s}}/r, p_{\mathrm{A}}) \cup \\ & S_{\mathrm{l}}(\beta i/r L_{\mathrm{s}} r, p_{\mathrm{C}}) \cup S_{\mathrm{r}}(B_{\mathrm{J}}, i L_{\mathrm{s}}/r, M) \end{aligned} \tag{3.68}$$

donde al vector de área de auto-interferencia se le añaden los de interferencia cruzada de las estructuras de datos `R2`, `B`, `WB`, `A`, `C` y `D`. Sumamos el área total de `WB` dado que se lo accede por completo en la línea 10; y lo mismo cabe decir de el bloque de `B` procesado. El modelado de los restantes vectores y matrices ya ha sido explicado.

Para calcular el vector de interferencia para cada línea desde el acceso en la línea 15 del programa en la iteración anterior del bucle en `K2` (línea 6) usamos una expresión independiente del índice de la línea, pues la distancia desde el último acceso en este bucle es la misma para todas las líneas:

$$\begin{aligned} S_{\text{int ext K2 RI}} =& S_{\text{s}}(C(Nr)C_{\text{sk}}) \cup S_{\text{s}}(Nr) \cup S_{\text{r}}(N_{B_{\text{J}}}, B_{\text{K}}, N) \cup \\ & S_{\text{s}}(B_{\text{J}}B_{\text{K}}) \cup S_{\text{l}}(N_{\text{nz}}, p_{\text{A}}) \cup S_{\text{l}}(N_{\text{nz}}r, p_{\text{C}}) \cup S_{\text{s}}(MB_{\text{J}}) \end{aligned} \tag{3.69}$$

que tiene en cuenta que desde el acceso a la línea considerada en el procesamiento del bloque anterior en K se ha recorrido la totalidad de los vectores `A` y `C` así como las porciones de las M filas de la matriz producto correspondientes a este bloque de la dimensión J.

Finalmente, entre dos accesos consecutivos a la línea de `R` en el bucle que nos ocupa, se genera un vector de área de interferencia cruzada independiente de la línea:

$$\begin{aligned} S_{\text{int I R}} =& S_{\text{s}}(B_{\text{K}}p_{\text{n}}) \cup S_{\text{s}}((B_{\text{K}}p_{\text{n}} + 1)r) \cup \\ & S_{\text{l}}(B_{\text{J}}B_{\text{K}}, p_{\text{WB}}) \cup S_{\text{r}}(B_{\text{J}}, 1, M) \end{aligned} \tag{3.70}$$

donde los términos corresponden, respectivamente, al acceso a los $B_{\text{K}}p_{\text{n}}$ elementos de la fila del bloque en el vector `A`, otros tantos en `C`, más el que se lee para detectar el final de la fila; el acceso con probabilidad uniforme a la matriz `WB` y finalmente, una fila de la matriz producto. De esta forma, el número de fallos sobre `R` en este acceso sería:

$$\begin{aligned} F_{\text{I R}} =& N_{B_{\text{J}}} \left(\sum_{i=1}^{N_{\text{l R}}} S_{\text{int ext c RI}_0}(i) + (N_{B_{\text{K}}} - 1) N_{\text{l R}} S_{\text{int ext K2 RI}_0} \right) + \\ & N_{B_{\text{J}}} N_{B_{\text{K}}} (M - N_{\text{l R}}) S_{\text{int I R}_0} \end{aligned} \tag{3.71}$$

en donde el último sumando se refiere a los accesos a los elementos que no son el primero de una línea. En el primero usamos $S_{\text{int ext c RI}_0}(i)$ para el primer acceso a cada línea en el primer bloque en la dimensión K y $S_{\text{int ext K2 RI}_0}$ para los restantes.

3.7.2. Vector R2

La diferencia entre el patrón de acceso a R y R2 es que este último es accedido tras la ejecución del bucle principal en I en la línea 30 (ver figura 3.13). Para modelar este hecho, calcularemos el número de fallos sobre R2 como:

$$F_{\mathrm{I\ R2}} = N_{B_\mathrm{J}} \left(\sum_{i=1}^{N_{\mathrm{l\ R}}} S_{\mathrm{int\ ext\ c\ RI}_0}(i) + (N_{B_\mathrm{K}} - 1) N_{\mathrm{l\ R}} S_{\mathrm{int\ ext\ K2\ RI}_0} \right) + N_{B_\mathrm{J}} N_{B_\mathrm{K}} M S_{\mathrm{int\ I\ R}_0} \tag{3.72}$$

La ecuación se explica teniendo en cuenta que el primer acceso a cada línea tiene el mismo comportamiento en ambos vectores, mientras que R2 tiene un nuevo acceso para cada elemento tras el cuerpo principal del bucle, cuyo vector de área de interferencia es $S_{\mathrm{int\ I\ R}}$. Por otra parte, los accesos a las últimas $L_\mathrm{s}/r - 1$ palabras de cada línea no tendrán probabilidad de fallo en el acceso de la línea 14 porque ésta ha sido accedida precisamente al final de la iteración anterior para el dato precedente.

3.7.3. Vector A

Los accesos a A, vector que contiene los valores reales no nulos de la matriz dispersa, son totalmente secuenciales dentro de la fila de cada bloque, pasando al terminar una fila de un bloque a la siguiente correspondiente al mismo bloque. Consideraremos cuatro distancias posibles en el reuso:

1. La que se da en dos iteraciones consecutivas del bucle en K de la línea 19. Es fácil comprobar que durante una iteración del bucle en K se genera un vector de área de interferencia cruzada:

$$S_{\mathrm{int\ K\ A}} = S_\mathrm{s}(B_\mathrm{J}) \cup S_\mathrm{s}(1) \tag{3.73}$$

   derivado de la lectura de B_J palabras consecutivas de WB y una línea de C.

2. La que se da en dos iteraciones consecutivas del bucle en I de la línea 13 si pueden compartir línea la última palabra leída en la iteración anterior y la primera de la actual. Habría de añadirse los accesos externos al bucle en K que se dan en una iteración del bucle en I, con lo que tendríamos:

$$S_{\mathrm{int\ I\ A}} = S_{\mathrm{int\ K\ A}} \cup S_\mathrm{s}(1) \cup S_\mathrm{s}(1) \cup S_\mathrm{r}(B_\mathrm{J}, 2, M) \tag{3.74}$$

por los accesos a un elemento de los vectores `R` y `R2` y dos filas de la matriz producto.

3. La que se da en dos iteraciones consecutivas del bucle en `K2` de la línea 6, pues en cada fila de cada nuevo bloque de la dimensión K el primer componente estará muy probablemente en la última línea accedida durante el procesamiento de la fila del bloque anterior. El vector de interferencia vendrá dado por:

$$\begin{aligned} S_{\text{int K2 A}} =& S_{\text{la}}(N_{\text{nz}}, p_{\text{A}}) \cup S_{\text{s}}(Mr) \cup S_{\text{s}}(Mr) \cup \\ & S_{\text{s}}(B_{\text{J}}B_{\text{K}}) \cup S_{\text{r}}(B_{\text{J}}, B_{\text{K}}, N) \cup \\ & S_{\text{s}}(MB_{\text{J}}) \cup S_{\text{l}}(N_{\text{nz}}r, p_{\text{C}}) \end{aligned} \tag{3.75}$$

en donde el vector de auto-interferencia para el acceso a `A` en una iteración del bucle `K2` tiene un comportamiento aproximado utilizando un acceso con probabilidad uniforme de acceso por línea p_{A}, como en la sección precedente. Las interferencias cruzadas vienen dadas, en este orden, por los accesos a las estructuras de datos `R`, `R2` y `WB` por completo, un bloque de la matriz `B`, B_{J} columnas de la matriz producto y el acceso a `C` modelado de forma similar al de `A`.

Cabe reseñar que en el caso de que $B_{\text{K}} = N$, $S_{\text{int K2 A}}$ no tendría sentido en sí mismo y debería adoptar en todo caso el mismo valor que $S_{\text{int J2 A}}$, que explicamos a continuación.

4. La que se da en dos iteraciones consecutivas del bucle en `J2` de la línea 1. La diferencia radica en que sabemos que tanto $\hat{\text{A}}$ como `C` se han accedido por completo, ya que se han completado las iteraciones del bucle en `K2`; y habremos leído B_{J} columnas completas de la matriz `B`, con lo cual:

$$\begin{aligned} S_{\text{int J2 A}} =& S_{\text{s}}(C(N_{\text{nz}})C_{\text{sk}}) \cup S_{\text{s}}(Mr) \cup S_{\text{s}}(Mr) \cup \\ & S_{\text{s}}(B_{\text{J}}B_{\text{K}}) \cup S_{\text{s}}(NB_{\text{J}}) \cup S_{\text{s}}(MB_{\text{J}}) \cup S_{\text{s}}(N_{\text{nz}}r) \end{aligned} \tag{3.76}$$

Para calcular el número de fallos sobre este vector estimaremos el número medio de fallos producidos en una iteración del bucle en `I` la línea 13 $F_{\text{I A}}$. El número total de fallos vendrá dado, pues, por:

$$F_{\text{A}} = N_{B_{\text{J}}} N_{B_{\text{K}}} M F_{\text{I A}} \tag{3.77}$$

Para estimar dicho número de fallos emplearemos $N_{\text{l I A}}$, el número medio de líneas del vector afectadas por los accesos durante una iteración del bucle mencionado:

$$N_{\text{l I A}} = \frac{B_{\text{K}} p_{\text{n}} + (L_{\text{s}} - 1)(1 - (1 - p_{\text{n}})^{B_{\text{K}}})}{L_{\text{s}}} \tag{3.78}$$

donde el numerador es el número medio de palabras del vector `A` traídas a la caché en una iteración de este bucle, como se comentó al explicar (3.63). En este caso no se multiplica el término de las $L_s - 1$ palabras adicionales por $(1 - p_{\text{comp A}})$ porque no se trata de considerar la superficie total afectada por el acceso considerando los precedentes sino el número de líneas a las que afecta el acceso por cada fila del bloque. La probabilidad de acierto para el acceso a la primera palabra de cada grupo se estima mediante:

$$\begin{aligned} P_{\text{ac A}} =& p_{\text{comp A}}(1 - S_{\text{int I A}_0}) + (1 - p_{\text{comp A}}) \\ & \left(\frac{(L_s - 1)N_{B_J}}{L_s N_{B_J}} (1 - S_{\text{int K2 A}_0}) + \frac{N_{B_J} - 1}{L_s N_{B_J}} (1 - S_{\text{int J2 A}_0}) \right) \end{aligned} \tag{3.79}$$

donde $p_{\text{comp A}}$, desarrollado en (3.64), es la probabilidad de que la última palabra de la fila precedente del bloque resida en la misma línea que la primera palabra de la fila actual, con lo que la probabilidad de fallo en ese caso es $S_{\text{int I A}_0}$. En caso contrario, la línea habrá sido accedida por última vez en la iteración precedente del bucle en `K2`, siempre que el bloque correspondiente no tuviese la última palabra de la fila en la última posición de una línea de caché. En este último caso, el primer acceso resultará en un fallo intrínseco y los $N_{B_J} - 1$ restantes tendrán como vector de interferencia cruzada $S_{\text{int J2 A}}$.

El primer elemento de las últimas $N_{\text{l I A}} - \text{mín}\{N_{\text{l I A}}, 1\}$ líneas no puede haber sido accedido en la iteración precedente del bucle de la línea 13, ni tampoco en la del bucle en `K2`, con lo que su probabilidad de fallo será $S_{\text{int J2 A}_0}$ para $N_{B_J} - 1$ de los bloques en la dimensión `J` (para el primero tendremos un fallo intrínseco). Finalmente, los elementos que no son el primero de una línea o el primer de todos en el grupo de elementos del vector `A` accedidos en una iteración del bucle en `I` de la línea 13, habrán de tener por vector de interferencia $S_{\text{int K A}}$. Por tanto $F_{\text{I A}}$ se estimará como:

$$\begin{aligned} F_{\text{I A}} =& (N_{\text{l I A}} - \text{mín}\{N_{\text{l I A}}, 1\}) \left(\frac{1}{N_{B_J}} + \frac{N_{B_J} - 1}{N_{B_J}} S_{\text{int J2 A}_0} \right) + \\ & \text{mín}\{N_{\text{l I A}}, 1\}(1 - P_{\text{ac A}}) + (B_K p_n - N_{\text{l I A}}) S_{\text{int K A}_0} \end{aligned} \tag{3.80}$$

3.7.4. Vector `C`

Observando el algoritmo puede comprobarse fácilmente que el acceso a `C` sigue un patrón muy similar al de `A`. Se distinguen en que `C` es un vector de enteros, por lo que debe utilizarse el factor de corrección r para calcular la superficie que ocupa cada bloque así como la distancia entre los bloques de dos filas consecutivas; y en que `C` tiene al menos un acceso obligatorio por

fila, puesto que es el que se chequea para saber antes de acceder a `A`, si el elemento correspondiente pertenece o no al bloque de columnas actual. Por este motivo, `C` tendrá $B_{\mathrm{K}}p_{\mathrm{n}}+1$ accesos en lugar de $B_{\mathrm{K}}p_{\mathrm{n}}$ por término medio durante el acceso a cada fila de cada bloque, siempre que haya más de un bloque en la dimensión K, pues en otro caso es imposible acceder a más de $B_{\mathrm{K}}p_{\mathrm{n}}$ elementos. De esta forma podemos estimar $N_{\mathrm{l\ I\ C}}$, el número de líneas afectadas por el acceso al vector `C` durante una iteración del bucle en `I` de la línea 13 como:

$$N_{\mathrm{l\ I\ C}} = \frac{(B_{\mathrm{K}}p_{\mathrm{n}} + \mathrm{Menor}(B_{\mathrm{K}}, N))r + L_{\mathrm{s}} - 1}{L_{\mathrm{s}}} \tag{3.81}$$

Usaremos también aquí los cuatro vectores de área de interferencia correspondientes a una iteración de los bucles de las líneas 19, 13, 6 y 1, definidos respectivamente sobre las variables `K`, `I`, `K2` y `J2`, al igual que se hizo en el apartado anterior. El cálculo es además idéntico al caso precedente, teniendo en cuenta que la auto-interferencia en $S_{\mathrm{int\ K2\ C}}$ vendrá dada por $S_{\mathrm{la}}(N_{\mathrm{nz}}r, p_{\mathrm{C}})$ y el vector de área de interferencia cruzada generada por el acceso a `A` en una iteración del bucle en `K2` será $S_{\mathrm{l}}(N_{\mathrm{nz}}, p_{\mathrm{A}})$. De forma similar, en el cálculo de $S_{\mathrm{int\ J2\ C}}$ el vector de área de auto-interferencia es $S_{\mathrm{s}}(C(N_{\mathrm{nz}}r)C_{\mathrm{sk}})$, y el vector de área de interferencia cruzada asociada al vector `A`, $S_{\mathrm{s}}(N_{\mathrm{nz}})$.

Por otra parte, al calcular la probabilidad de acierto en el primer acceso al grupo de entradas correspondientes a una fila del bloque debemos tener en cuenta que en la lectura de la misma fila para el bloque previo en la dimensión K, que es la que afecta a este vector, se ha leído la primera palabra asociada al bloque actual. Por tanto dicha probabilidad de acierto tiene la expresión:

$$P_{\mathrm{ac\ C}} = p_{\mathrm{comp\ C}}(1 - S_{\mathrm{int\ I\ C_0}}) + (1 - p_{\mathrm{comp\ C}})(1 - S_{\mathrm{int\ K2\ C_0}}) \tag{3.82}$$

Por otra parte, la seguridad de que hay al menos un acceso por fila de cada bloque simplifica también la expresión del número de fallos durante una iteración del bucle en `I` con respecto a la correspondiente para el vector `A`:

$$\begin{aligned} F_{\mathrm{I\ C}} =& (N_{\mathrm{l\ I\ C}} - 1)\left(\frac{1}{N_{B_{\mathrm{J}}}} + \frac{N_{B_{\mathrm{J}}} - 1}{N_{B_{\mathrm{J}}}} S_{\mathrm{int\ J2\ C_0}}\right) + \\ & (1 - P_{\mathrm{ac\ C}}) + (B_{\mathrm{K}}p_{\mathrm{n}} + \mathrm{Menor}(B_{\mathrm{K}}, N) - N_{\mathrm{l\ I\ C}})S_{\mathrm{int\ K\ C_0}} \end{aligned} \tag{3.83}$$

3.7.5. Vector `D`

Este vector se accede por bloques de B_{J} elementos consecutivos dentro de una fila en cada iteración del bucle `I` de la línea 13. Estos accesos tienen

la forma de una lectura consecutiva justo antes de entrar en el bucle más interno en `K` y una escritura al salir. Calcularemos el número medio de fallos $F_{\mathrm{L\ D}}$ durante la lectura y de fallos en la escritura $F_{\mathrm{E\ D}}$ para cada bloque de B_{J} columnas, de forma que el número total de fallos sobre esta matriz se obtendrá como:

$$F_{\mathrm{D}} = N_{B_{\mathrm{J}}}(F_{\mathrm{L\ D}} + F_{\mathrm{E\ D}}) \tag{3.84}$$

Un grupo de B_{J} columnas de la matriz producto constará de MB_{J} elementos consecutivos y por tanto, de $N_{\mathrm{l\ D}} = MB_{\mathrm{J}}/L_{\mathrm{s}}$ líneas. Cada grupo es leído $N_{B_{\mathrm{K}}}$ veces, de las cuales en la primera tendremos un fallo intrínseco por línea, y en las restantes se habrá de considerar el vector de interferencia generado durante una iteración completa del bucle en `K2`:

$$\begin{aligned} S_{\mathrm{int\ K2\ D}} =& S_{\mathrm{s}}(C(MB_{\mathrm{J}})C_{\mathrm{sk}}) \cup S_{\mathrm{s}}(B_{\mathrm{J}}B_{\mathrm{K}}) \cup S_{\mathrm{r}}(B_{\mathrm{J}}, B_{\mathrm{K}}, N) \cup \\ & S_{\mathrm{s}}(Mr) \cup S_{\mathrm{s}}(Mr) \cup S_{\mathrm{l}}(N_{\mathrm{nz}}, p_{\mathrm{A}}) \cup S_{\mathrm{l}}(N_{\mathrm{nz}}r, p_{\mathrm{C}}) \end{aligned} \tag{3.85}$$

en donde comenzamos con el vector de área de auto-interferencia y añadimos los de interferencia cruzada asociados a `WB`, un bloque de la matriz `B`, `R`, `R2` y las porciones de los vectores `A` y `C` afectadas por los accesos durante el procesamiento de un bloque de la dimensión K. Todos ellos han sido explicados ya en apartados anteriores.

En referencia a los elementos que no son el primero de su línea en ser accedidos durante una iteración del bucle en `K2`, tendrán como vector de interferencia asociado:

$$S_{\mathrm{int\ I\ D}} = S'_{\mathrm{ar}}(B_{\mathrm{J}}, 1, M) \cup S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(2) \tag{3.86}$$

puesto que a un elemento de una línea perteneciente a una columna dada le afectan las auto-interferencias derivadas de la escritura de las líneas de las columnas con índice mayor que la suya para la fila precedente y la lectura de las asociadas a las columnas con índice inferior para la fila actual, situación descrita en la sección 2.3.5. Por otra parte, tenemos el acceso a un elemento de `R` y dos de `R2`.

El número de fallos en las lecturas se estimará, pues, como:

$$F_{\mathrm{L\ D}} = N_{\mathrm{l\ D}} + (N_{B_{\mathrm{K}}} - 1)N_{\mathrm{l\ D}}S_{\mathrm{int\ K2\ D}_0} + N_{B_{\mathrm{K}}}(MB_{\mathrm{J}} - N_{\mathrm{l\ D}})S_{\mathrm{int\ I\ D}_0} \tag{3.87}$$

En cuanto a las escrituras, entre la lectura de una fila del bloque y su escritura sólo encontramos el bucle en `K`, así pues todas las palabras implicadas tienen el mismo vector de área de interferencia, que viene dado por el

vector de área de auto-interferencia del acceso a la fila mencionada más los de interferencia cruzada asociados a los datos accedidos en dicho bucle:

$$\begin{aligned} S_{\text{int K D}} =& S_{\text{ar}}(B_{\text{J}}, 1, M) \cup S_{\text{l}}(B_{\text{J}} B_{\text{K}}, p_{\text{WB}}) \cup \\ & S_{\text{s}}(B_{\text{K}} p_{\text{n}}) \cup S_{\text{s}}((B_{\text{K}} p_{\text{n}} + \text{Menor}(B_{\text{K}}, N)) r) \end{aligned} \tag{3.88}$$

en donde el valor de la probabilidad aproximada de acceso por línea de `WB` se ha calculado en (3.61), y se acceden aproximadamente a $B_{\text{K}} p_{\text{n}}$ elementos de `A` y otros tantos de `C`, más uno si hay varios bloques. Por tanto $F_{\text{E D}}$ viene dado por:

$$F_{\text{E D}} = M B_{\text{J}} N_{B_{\text{K}}} S_{\text{int K D}_0} \tag{3.89}$$

3.7.6. Vector `B`

Cada elemento del vector `B` es accedido una y sólo una vez, realizándose este acceso en el momento de copiar dicho elemento al área de trabajo definida por el vector `WB`. Este proceso de copiado se realiza a nivel de bloques, una vez por cada bloque, con lo que tiene lugar en total $N_{B_{\text{J}}} N_{B_{\text{K}}}$ veces. La copia del bloque se realiza accediendo `B` por columnas de B_{K} elementos cada una. Por tanto, si calculamos el número de fallos durante la copia de dichos elementos consecutivos y la multiplicamos por $N_{B_{\text{J}}} N_{B_{\text{K}}} B_{\text{J}} = N_{B_{\text{K}}} H$ (puesto que cada bloque tiene B_{J} columnas) obtendremos el número total de fallos en `B`. Dicho valor corresponderá a un fallo intrínseco por línea y un vector de interferencia cruzada generado por el acceso a un elemento de `WB` ($S_{\text{s}}(1)$), con lo cual tenemos que el número de fallos sobre la matriz `B` viene dado por:

$$F_{\text{B}} = N_{B_{\text{K}}} H \left(\frac{BK + L_{\text{s}} - 1}{L_{\text{s}}} + \left(B_{\text{K}} - \frac{BK + L_{\text{s}} - 1}{L_{\text{s}}} \right) S_{\text{s}_0}(1) \right) \tag{3.90}$$

3.7.7. Vector `WB`

El vector `WB` contiene una copia del bloque actual de la matriz `B` traspuesto, por lo que consta de B_{J} filas y B_{K} columnas. El proceso de copia se realiza una vez por bloque, empleándose el vector `WB` en el bucle más interno, donde se accede por columnas. Calcularemos por una parte los fallos producidos en en los accesos a `WB` en el bucle más interno, y por otro, los que se ocasionan durante la copia de `B` a `WB` en la línea 10.

Fallos en el bucle interno

La probabilidad de acierto en un acceso a una línea dada de la matriz `WB` durante el procesamiento de la j-ésima fila de la matriz dispersa para un bloque dado es:

$$\begin{aligned} P_{\text{ac I WB}}(j) = &\sum_{i=1}^{j-1} p_{\text{WB}}(1-p_{\text{WB}})^{i-1}(1-S_{\text{int I WB}_0}(i)) \\ &+ (1-p_{\text{WB}})^{j-1}(1-S_{\text{int I WB}_0}(j))(1-S_{\text{int init WB}_0}) \end{aligned} \tag{3.91}$$

en donde se recordará que p_{WB}, desarrollada en (3.61), es la probabilidad de acceder a una línea de la matriz `WB` durante el procesamiento de una fila de un bloque de la matriz dispersa. De esta forma, $p_{\text{WB}}(1-p_{\text{WB}})^{i-1}$ en (3.91) es la probabilidad de que el último acceso a la línea que estamos considerando se haya producido durante el procesamiento de la fila $j-i$ de la matriz dispersa. Por otro lado, $S_{\text{int I WB}}(i)$ es el vector de área correspondiente a los accesos a todos los vectores involucrados en el procesamiento de i filas del bloque de la matriz dispersa durante las correspondientes i iteraciones del bucle en `I` de la línea 13. El valor de este vector viene dado por:

$$\begin{aligned} S_{\text{int I WB}}(i) = &S^i_{\text{la}}(B_{\text{J}}B_{\text{K}}, p_{\text{WB}}) \cup S_{\text{s}}(ir) \cup S_{\text{s}}(ir) \cup \\ &S_{\text{l}}(\beta i, p_{\text{A}}) \cup S_{\text{l}}(\beta ir, p_{\text{C}}) \cup S_{\text{r}}(B_{\text{J}}, i, M) \end{aligned} \tag{3.92}$$

en donde al vector de área de auto-interferencia para los $B_{\text{J}}B_{\text{K}}$ elementos de la matriz `WB`, con probabilidad uniforme de acceso por línea $1-(1-p_{\text{WB}})^i$ añadimos, en este orden, los vectores de área de interferencia cruzada de `R`, `R2`, `A`, `C` y la matriz producto `D`.

Así, el sumatorio de (3.91) tiene la misma forma, por ejemplo, que el de la expresión (3.10), correspondiente al modelado del vector `X` en el producto matriz dispersa-vector, al tratarse del mismo tipo de patrón de acceso. Sin embargo, en este caso existe una probabilidad de acierto en el primer acceso debida al acceso previo a `WB` en la copia del bloque de la matriz `B`. De ahí la existencia del sumando que constituye la segunda línea de la fórmula: multiplicamos la probabilidad de que la línea no haya sido accedida en las iteraciones previas del bucle en `I` (probabilidad $(1-p_{\text{WB}})^{j-1}$) por la probabilidad de que no haya sido expulsada por las interferencias a lo largo de dichas iteraciones $(1-S_{\text{int I WB}_0}(j))$ y la probabilidad de que la línea se encontrase en la caché al iniciarse la ejecución del bucle $(1-S_{\text{int init WB}_0})$. El vector de área de interferencia medio para una línea de `WB` desde su último acceso en la línea 10 hasta que se inicia el bucle en `I` que estamos considerando ha sido

estimado como:

$$S_{\text{int init WB}} = S_{\text{s}}(C(B_{\text{J}}B_{\text{K}})C_{\text{sk}}/2) \cup S_{\text{B}} \tag{3.93}$$

donde el primer término es el vector de auto-interferencia medio del vector consigo mismo y el segundo da el vector medio de interferencia cruzada de los accesos a la matriz `B`. Estudiando el bucle que comprende las líneas 8–12, vemos que si $B_{\text{J}} \leq L_{\text{s}}$, en cada iteración del bucle en `J` de la línea 8 se accede a todas las líneas de `WB`. Por tanto, la interferencia media generada por los accesos a `B` para una línea cualquiera de `WB` sería $S_{\text{s}}(B_{\text{K}}/2)$. En caso contrario, podemos aproximar la interferencia media de `B` como $S_{\text{r}}(B_{\text{J}}/2, B_{\text{K}}, N)$, es decir, el acceso a $B_{\text{J}}/2$ columnas del bloque de `B`.

El número de fallos sobre la matriz `WB` en este bucle vendrá dado por el producto de tres términos: la probabilidad media de fallo, el número medio de líneas accedidas cuando se lee una columna de esta matriz y el número de veces que esta lectura tiene lugar, es decir, una por entrada y por bloque en la dimensión J:

$$F_{\text{I WB}} = \left(1 - \frac{\sum_{j=1}^{m} P_{\text{ac I WB}}(j)}{M}\right) \text{Grupos}(L_{\text{s}}, B_{\text{J}}) N_{\text{nz}} N_{B_{\text{J}}} \tag{3.94}$$

Fallos en la copia

Calcularemos el número de fallos en la copia multiplicando la estimación del número de fallos por proceso de copia por el número de bloques $N_{B_{\text{J}}} N_{B_{\text{K}}}$. La media de fallos durante una copia se estima como:

$$F_{\text{cop WB}} = \sum_{j=1}^{B_{\text{J}}} \sum_{i=1}^{\frac{B_{\text{J}}B_{\text{K}}}{L_{\text{s}}}} A \cdot S_{\text{s}_0}(1) + \left(\frac{L_{\text{s}}}{B_{\text{J}}} - A\right)(1 - P_{\text{ac cop WB}}(i,j)) \tag{3.95}$$

en donde $A = \text{máx}\{0, \frac{L_{\text{s}}}{B_{\text{J}}} - 1\}$ es el número medio de accesos tras el primero a una línea dada de la matriz `WB` durante una iteración del bucle de la línea 8. Como muestra la ecuación, la probabilidad de fallo para estos accesos es la asociada a la lectura de un elemento de la matriz `B`. La probabilidad de acierto para el primer acceso a la i-ésima línea durante la iteración j del bucle, $P_{\text{ac cop WB}}(i,j)$, se calcula como:

$$\begin{aligned} P_{\text{ac cop WB}}(i,j) =& P_{\text{acc}}(j-1)(1 - (S'_{\text{ra}}(B_{\text{K}}, 1, B_{\text{J}}) \cup S_{\text{s}}(B_{\text{K}}))_0) + \\ & (1 - P_{\text{acc}}(j-1)) \frac{N_{B_{\text{J}}} N_{B_{\text{K}}} - 1}{N_{B_{\text{J}}} N_{B_{\text{K}}}} (1 - (S_{\text{I WB}} \cup S_{\text{exp WB}}(i,j))_0) \end{aligned} \tag{3.96}$$

donde $P_{\text{acc}}(j)$ es la probabilidad de que la línea haya sido accedida en las j iteraciones anteriores. Su valor viene dado por:

$$P_{\text{acc}}(j) = \begin{cases} \text{mín}\{1, (L_{\text{s}} + j - 1)/B_{\text{J}}\} & \text{si } j > 0 \\ 0 & \text{si } j = 0 \end{cases} \tag{3.97}$$

El primer acceso a una línea resulta en un fallo a no ser que se encuentre en la caché cuando comienza la copia y que no haya sido reemplazada durante la copia de los elementos accedidos previamente. Si $S_{\text{I WB}}$ es el vector de área de interferencia medio generado desde el último acceso a la línea considerada hasta que finaliza el bucle en `I` estudiado anteriormente y $S_{\text{exp WB}}(i, j)$ es el vector de área de interferencia para la linea i-ésima de `WB` durante j iteraciones del bucle de la línea 8, su unión será el vector de área de interferencia total para este caso. Esto será cierto para todos los bloques excepto el primero, donde tendremos fallos intrínsecos, de ahí la multiplicación por $(N_{B_{\text{J}}} N_{B_{\text{K}}} - 1)/(N_{B_{\text{J}}} N_{B_{\text{K}}})$. Por otro lado, si la línea ha sido accedida en la iteración previa, únicamente los accesos a B_{K} elementos de la matriz `B` localizados en dos columnas consecutivas (cuya área aproximamos mediante un acceso completamente secuencial) y los accesos requeridos para finalizar el procesamiento de la fila anterior de la matriz `WB` y comenzar la de la actual pueden haber reemplazado la línea. En cuanto a $S_{\text{exp WB}}(i, j)$ es estimado como la unión de los siguientes elementos:

- $S_{\text{s}}(i \cdot L_{\text{s}}/B_{\text{J}})$, correspondiente a $iL_{\text{s}}/B_{\text{J}}$ elementos consecutivos de la columna de la matriz `B` que se está copiando.
- $S_{\text{r}}(j - 1, B_{\text{K}}, N)$, asociado a $j - 1$ columnas del bloque de la matriz `B` con B_{K} elementos cada una.
- $S_{\text{l}}(\lfloor (iL_{\text{s}} - 1)/C_{\text{sk}} \rfloor C_{\text{sk}}, P_{\text{acc}}(j))$, que es el vector de área de auto-interferencia para las líneas de la matriz `WB` con índice inferior a i.
- $S_{\text{l}}(\lfloor (B_{\text{J}} B_{\text{K}} - iL_{\text{s}})/C_{\text{sk}} \rfloor C_{\text{sk}}, P_{\text{acc}}(j-1))$, que representa el vector de área de auto-interferencia con las líneas de la matriz con índice mayor que i.

Por último, el valor de $S_{\text{I WB}}$ puede calcularse como una media de los vectores de interferencia que participan en (3.91) para $j = M + 1$:

$$\begin{aligned} S_{\text{I WB}} = & \sum_{i=1}^{M} p_{\text{WB}} (1 - p_{\text{WB}})^{i-1} S_{\text{int I WB}}(i)) + \\ & (1 - p_{\text{WB}})^{M} (S_{\text{int I WB}}(M)) \cup S_{\text{int init WB}}) \end{aligned} \tag{3.98}$$

N	N_{nz}	p_{n}	H	B_{J}	B_{K}	C_{s}	L_{s}	K	σ	Δ
1.5	125	5.5 %	1.5	25	500	8	4	1	0.55	2.76
1.5	125	5.5 %	1.5	25	500	8	4	4	0.26	4.13
1.5	125	5.5 %	1.5	25	500	16	4	2	0.46	0.15
1.5	125	5.5 %	1.5	25	500	16	4	4	1.09	-2.09
5	500	2.0 %	0.1	10	500	4	8	2	0.45	2.36
5	500	2.0 %	0.1	10	1000	4	8	2	0.25	3.13
5	500	2.0 %	0.1	24	2100	16	8	1	0.46	8.97
5	500	2.0 %	0.1	26	2100	16	8	1	0.27	10.38
5	500	2.0 %	0.1	24	2100	16	8	2	0.08	8.40
5	500	2.0 %	0.1	28	2100	16	8	2	0.10	10.13
10	100	0.1 %	10	20	1000	16	4	1	0.16	2.39
10	100	0.1 %	10	20	1000	16	16	2	0.12	-1.62
10	100	0.1 %	10	10	1000	16	16	4	0.08	-1.26
10	100	0.1 %	10	10	1000	16	4	4	0.08	1.20
10	100	0.1 %	10	20	1000	32	8	2	0.41	1.48

Cuadro 3.7: Desviación del modelo para el producto matriz dispersa-matriz densa optimizado.

3.7.8. Validación y análisis

Este algoritmo fue validado mediante simulaciones sobre unas tres mil combinaciones de sus parámetros de entrada en las que se obtuvo una media para el valor absoluto del error cometido por el modelo del 2.36 %, y siendo la desviación típica media de las ejecuciones efectuadas para cada combinación de los parámetros probada del 0.70 % del número de fallos medidos. La tabla 3.7 muestra los datos de validación de algunas combinaciones. Puede observarse que las mayores desviaciones se dan para los bloques cuyas dimensiones no son divisores de las dimensiones de la matriz densa y que dan lugar a pocos bloques en esa dimensión, como cabría esperar. El uso de un tamaño de bloque 2100 en una dimensión con 5000 elementos genera dos bloques de 2100 componentes y uno de 800 en lugar de los 3 bloques de 2100 elementos que nuestro programa modela para simplificar el problema. El error es, pues, positivo (nuestro modelo estima más fallos de los que hay realmente) y elevado en relación con los observados para los valores de B_{K} divisores de N. El que B_{J} no sea divisor de H no afecta tanto debido al pequeño valor que tiene y al elevado número de bloques a que da lugar.

Dado que este algoritmo incluye el uso de bloques, hemos empleado en esta ocasión nuestro modelo para derivar el tamaño óptimo del bloque para

la ejecución de este código en varias plataformas. Los parámetros extraídos de cada sistema han sido su número de niveles N_{Niveles} junto con el tamaño de cada nivel, C_{s_i}, su tamaño de línea, L_{s_i}, su asociatividad, K_i, y, finalmente, su penalización por fallo $T_{\text{fallo}i}$. Los datos de la configuración de la jerarquía de memoria se han obtenido de la documentación en línea disponible para los sistemas probados, mientras que el peso relativo de la penalización en cada nivel ha sido extraído experimentalmente a partir del tiempo de ejecución de programas que generaban un número predeterminado de fallos en los diferentes niveles de la caché. La política seguida para compara los diferentes bloques ha consistido en calcular el coste de cada bloque como:

$$\text{Coste}(BJ, BK) = \sum_{i=1}^{N_{\text{Niveles}}} T_{\text{fallo}i} \cdot F(BJ, BK, C_{\text{s}_i}, L_{\text{s}_i}, K_i) \tag{3.99}$$

en donde $F(BJ, BK, C_{\text{s}}, L_{\text{s}}, K)$ proporciona el número total de fallos predichos por el modelo para este bloque y esta configuración de caché. Los parámetros que describen las matrices han sido eliminados para simplificar la expresión.

Las plataformas utilizadas para realizar el estudio fueron las siguientes: una estación DEC 433au Personal Workstation, basada en un procesador Alpha 21164 [6] a 433 MHz, un servidor SGI Origin 200 con procesadores R10000 [35] a 180 MHz y una estación SUN Ultra 1 con un procesador UltraSPARC-I [25] a 167 MHz. Creemos que representan un rango amplio de arquitecturas actuales adecuado para la validación de nuestros experimentos. La tabla 3.8 muestra los resultados obtenidos para varias matrices. El conjunto de prueba para B_{J} (tamaño del bloque en la dimensión J), fue 8, 10, 12, 16 y 20, mientras que los valores probados para B_{K} (tamaño del bloque en la dimensión K) fueron 50, 100, 200, 250, 500, 1000, 1250, 2500 y 5000 (estos últimos sólo cuando eran aplicables). La tabla muestra la diferencia entre los tiempos de ejecución obtenidos usando el bloque propuesto por el modelo, es decir, el que minimiza $\text{Coste}(BJ, BK)$, y el bloque óptimo, es decir, el que lleva al menor tiempo de ejecución. Esta diferencia se expresa como un porcentaje del tiempo obtenido para el bloque óptimo.

El modelo ha sido diseñado inicialmente para cachés de postescritura, considerando el mismo coste y comportamiento para un fallo de lectura y uno de escritura. Se ha desarrollado una variante para adaptarlo a cachés de escritura directa sin asignación, dado que el primer nivel de caché de los procesadores 21164 y UltraSPARC I sigue esta política. Ha consistido en considerar cualquier escritura en estas cachés como un fallo, pero dándole un peso proporcional al tiempo requerido para resolverlo. Además los accesos de

Matriz			Sistema		
M	N	p_{n}	433au	Origin 200	Ultra 1
1	1	0.05	0.00 %	11.46 %	0.00 %
1	1	0.10	0.00 %	10.64 %	0.00 %
2.5	2.5	0.08	2.10 %	3.28 %	0.19 %
2.5	2.5	0.16	0.00 %	2.75 %	0.00 %
5	2.5	0.01	6.83 %	0.00 %	1.96 %
5	5	0.04	0.00 %	0.00 %	9.98 %

Cuadro 3.8: Desviación del tiempo de ejecución usando el bloque propuesto por el modelo como porcentaje del tiempo óptimo de ejecución. M y N en miles, $H = 1000$.

escritura a estas cachés no generan vectores de área de interferencia, dado que los datos afectados no son traídos a la caché.

Puede verse que las peores estimaciones tienen lugar en el caso del servidor Origin 200 usando una pequeña matriz de orden mil. Aunque la diferencia porcentual es notable, el tiempo de ejecución del bloque propuesto por el modelo es menos de 0.1 segundos mayor que el óptimo para $p_{\mathrm{n}} = 0{,}05$ y 0.18 segundos para $p_{\mathrm{n}} = 0{,}1$. No obstante, hemos utilizado la herramienta `perfex` del sistema operativo a fin de acceder a los contadores internos disponibles en el R10000, los cuales pueden proporcionar informaciones tales como el número de fallos de datos en cada uno de los dos niveles de caché de que dispone su jerarquía de memoria.

El número de fallos medido no está próximo al predicho por el modelo, lo cual no es sorprendente, dado que hay varios efectos que éste no considera: la compartición del segundo nivel de la caché con el código; los patrones de acceso no son exactamente los esperados debido al uso de variables intermedias y a las optimizaciones del compilador, así como la medida de fallos de datos que el modelo no tiene en cuenta, tales como los que se producen cuando se lee la matriz dispersa del disco. Además la CPU está siendo compartida con otros procesos, los cuales generan fallos en los cambios de contexto.

De todas formas esperamos que los patrones más importantes sean los que refleja el modelo, siendo nuestro propósito proporcionar no una estimación cuantitativa exacta del número real de fallos, sino una idea correcta de la evolución del comportamiento de la caché con respecto a las variaciones de los parámetros de entrada del modelo, usando el número de fallos predicho como indicador de este comportamiento. Las mediciones obtenidas utilizando los contadores del procesador muestran que este objetivo se ha conseguido,

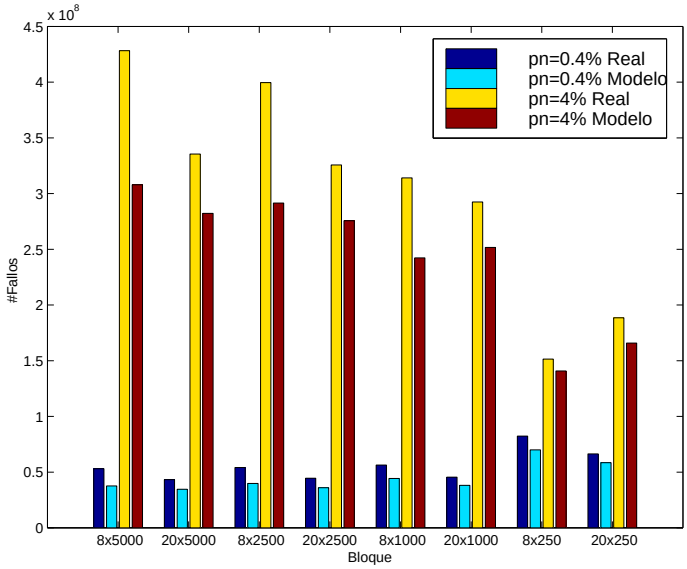

Figura 3.15: Número de fallos medidos y predichos en la caché de datos del primer nivel del procesador R10000 durante el producto de una matriz dispersa 5000x5000 con $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$ para diferentes bloques.

dado que casi siempre hay una proporcionalidad entre el número real de fallos medidos en cada nivel de la caché y el número de fallos que el modelo predice para él. Este hecho se refleja en la figura 3.15.

El bloque propuesto por el modelo es realmente más efectivo que el bloque óptimo desde el punto de vista de la jerarquía de memoria. Lo que ocurre es que el tiempo total de ejecución depende de más factores: el bloque propuesto por el modelo se adecua mejor a las cachés, dado que es menor que el bloque óptimo, pero su uso genera una carga mayor de operaciones al requerir más iteraciones en los bucles que controlan los bloques. Por otra parte, estamos hablando de procesadores superescalares capaces de iniciar múltiples instrucciones simultáneamente y de ocultar las latencias de memoria mediante varias estrategias, siendo precisamente ésta una de las características más destacables del R10000 [51]. Esto hace que el número de fallos sólo sea una aproximación cualitativa del rendimiento del programa.

Se pueden dar razones similares para las desviaciones observadas en la tabla 3.8 para la mayor matriz en el sistema Ultra 1. Además en este caso tenemos el inconveniente de que en la caché de datos del primer nivel cada línea está dividida en dos sub-bloques y nuestro modelo no soporta asignación por sub-bloques. Si bien se podría extender el modelo para soportar esta

técnica, creemos que no ha alcanzado una amplia aplicación en las cachés actuales por el momento.

El máximo valor para B_J empleado en estos experimentos ha sido 20 porque, como hemos explicado, debe haber B_J registros en punto flotante libres para implementar el *blocking* a nivel de registros y el compilador de FORTRAN para la 433au Personal Workstation utiliza hasta 19 registros para este propósito. Los tamaños de bloque óptimos han usado casi siempre el valor máximo de B_J en las tres máquinas probadas. Esto tiene sentido, dado que lleva al menor número de bloques en la dimensión J. Además se reduce el número de accesos y se favorece la explotación de la localidad espacial. El acceso a la matriz `WB` en el bucle más interno tiene lugar secuencialmente en esta dimensión, cuyos componentes se almacenan en posiciones de memoria consecutivas. Sin embargo, ha habido fuertes variaciones en los valores óptimos de B_K para las diferentes arquitecturas y matrices, lo cual nos ha llevado a hacer un estudio de influencia de los diferentes parámetros en el comportamiento de la caché.

En las figuras 3.16 a 3.18 hemos comprobado el comportamiento de la caché durante el producto de una matriz dispersa 5000x5000 como función de B_K manteniendo B_J constante e igual a 20. Cada una de las tres figuras considera diferentes niveles de asociatividad, tamaños de línea y de caché, respectivamente. Se han elegido valores típicos de cachés de segundo nivel para los parámetros de caché. En lo que sigue, el comportamiento de la caché se explica basándose sólo en la matriz `WB`, ya que generalmente es la que provoca la mayoría de los fallos.

La figura 3.16 muestra que los valores pequeños de p_n favorecen el uso de bloques grandes independientemente del tamaño de los conjuntos de la caché porque todos los tamaños de bloque considerados caben en la caché y hay pocas interferencias cruzadas debido al reducido número de entradas. Sin embargo, a medida que p_n crece, el tamaño del bloque óptimo se reduce debido al aumento del número de interferencias cruzadas. Además, tal como se esperaba, a menor valor de K, menor es el tamaño del bloque óptimo, dado que el efecto de las interferencias es mayor.

La influencia del tamaño de línea sobre el número de fallos se muestra en la figura 3.17. Las referencias dentro del bucle interno son básicamente 20 accesos de lectura a posiciones de memoria consecutivas, así pues, para tamaños de línea típicos para cachés de segundo nivel, a mayor línea, menor número de fallos. Sólo ante tamaños extremadamente grandes (≥ 256) comienza a darse un aumento del número de fallos debido a su impacto negativo sobre la probabilidad de interferencia. Por otra parte, el tamaño óptimo del bloque

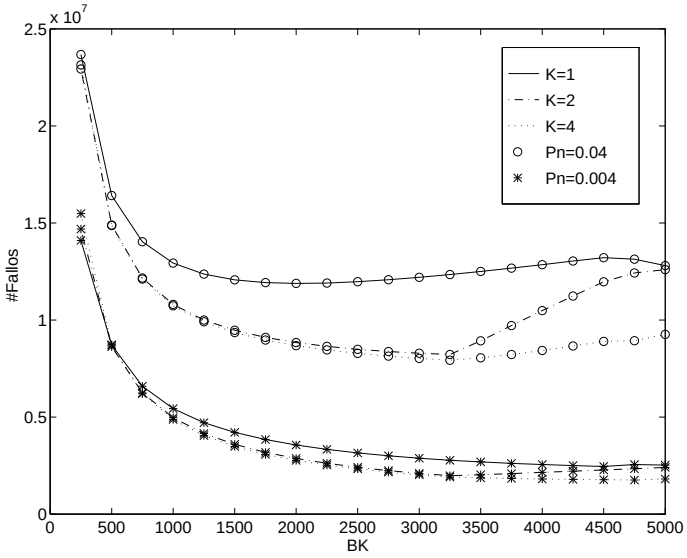

Figura 3.16: Número de fallos durante el producto de una matriz dispersa 5000x5000 para diferentes niveles de asociatividad en una caché de 128K palabras con un tamaño de línea de 16 palabras; $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$.

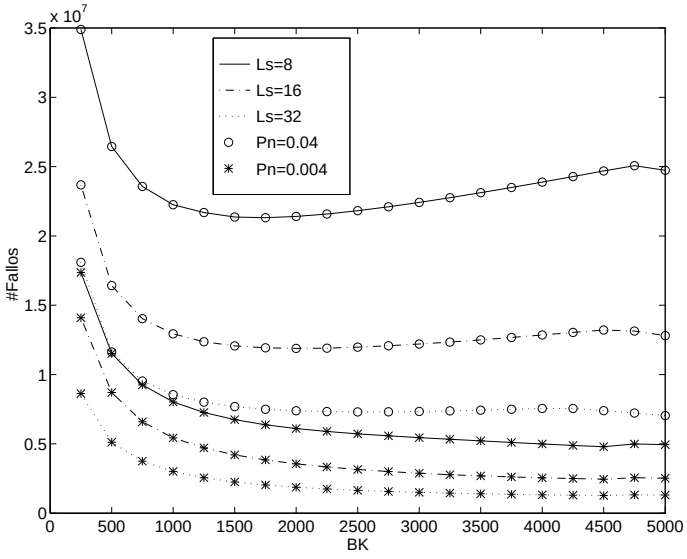

Figura 3.17: Número de fallos durante el producto de una matriz dispersa 5000x5000 para diferentes tamaños de línea en una caché de correspondencia directa de 128K palabras; $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$.

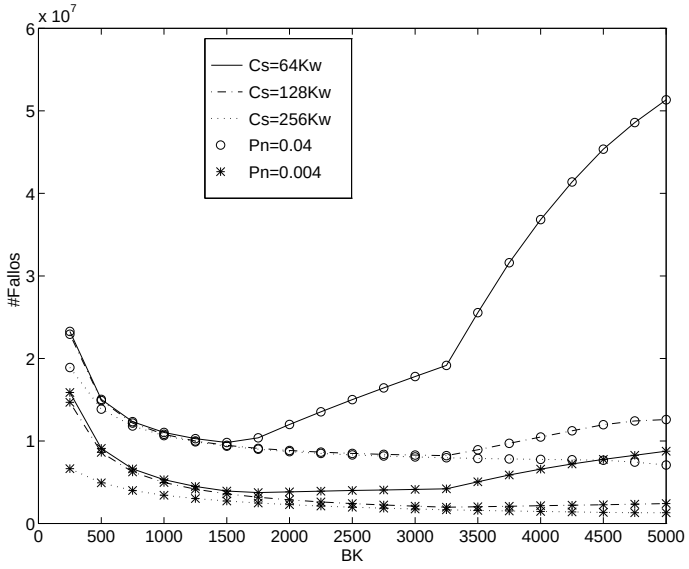

Figura 3.18: Número de fallos durante el producto de una matriz dispersa 5000x5000 para diferentes tamaños de caché en una caché de dos vías con una línea de 16 palabras; $p_{\mathrm{n}} = 0{,}004$ y $p_{\mathrm{n}} = 0{,}04$ usando $H = 1000$.

experimenta pocos cambios con respecto a L_{s}. Aunque no se muestra en el gráfico, el aumento de la asociatividad da lugar a un aumento del tamaño del bloque óptimo con el tamaño de línea, dado que equilibra el aumento de las interferencias que tiene lugar con los tamaños de línea mayores. Debe tenerse en cuenta que el aumento del tamaño de la línea puede llevar a un aumento del tiempo de ejecución a pesar de la reducción del número de fallos debido a la mayor penalización por fallo.

El comportamiento del tamaño del bloque óptimo en función del tamaño de la caché es escalable para los dos valores de p_{n} en la figura 3.18. Para $C_{\mathrm{s}} \geq 128$K palabras cualquier bloque cabe en la caché, con lo que el mayor bloque es el mejor para las cachés grandes. Sin embargo podemos apreciar el efecto observado en la figura 3.16 consistente en la reducción del tamaño del bloque óptimo a medida que p_{n} crece debido al aumento de la probabilidad de interferencia cruzada. El comportamiento de la única caché de segundo nivel considerada en la figura 3.18 para la que algunos bloques no caben, esto es, la de 64K palabras, limita el tamaño del bloque óptimo a ser el mayor inferior a C_{s} en el caso de $p_{\mathrm{n}} = 0{,}004$ a fin de evitar auto-interferencias y explotar la caché al máximo. A medida que p_{n} aumenta este límite se reduce debido al efecto combinado de las auto-interferencias y las interferencias cruzadas. La forma de la curva para la caché de 64K palabras y la matriz con $p_{\mathrm{n}} = 0{,}04$

muestra claramente la importancia relativa de las interferencias cruzadas y de las auto-interferencias: estas últimas son mucho más importantes.

3.8. Tiempos de simulación versus tiempos de modelado

Como se ha dicho en el capítulo introductorio, una de las ventajas del modelado analítico sobre la simulación es el gran ahorro de costos computacionales que supone. A fin de ilustrar esta diferencia en el caso concreto de nuestro modelo, presentamos en la tablas 3.9 a 3.11 el tiempo requerido (en segundos) para efectuar una simulación y el correspondiente cálculo para el modelado para varios algoritmos. En el caso de la simulación hemos usado el simulador de caché desarrollado por nosotros. El tiempo de ejecución es mucho menor que el empleado por un simulador conocido como dineroIII [32], al no requerir la generación y paso de la traza a un formato de texto e incluir menos parámetros para la configuración de la caché. A fin de comprobar esta diferencia también hemos incluido en la tabla 3.10 los tiempos de simulación correspondientes a dineroIII. Dicho tiempo no incluye el de generación de la traza. Todos los tiempos han sido obtenidos en un servidor SGI Origin 200 con procesadores MIPS R10000 a 180MHz.

En la tabla 3.10 puede apreciarse que para un algoritmo con un número de accesos relativamente reducido pero con un modelo complejo en relación a otros, el simulador simplificado puede dar resultados mejores que el modelo. No obstante, hemos de recordar que el modelo proporciona un valor medio para el número de fallos independiente de las posiciones relativas en que se encuentran las estructuras de datos. La simulación, por el contrario, sólo puede darnos un valor concreto, y como hemos visto en las tablas de validación, el número de fallos puede variar mucho con este parámetro. Por ello, sería necesaria la realización de un mayor número de simulaciones para estimar un número medio de fallos, perdiéndose de esta forma la ventaja que a primera vista podría proporcionar la simulación frente al modelado con respecto a los tiempos de ejecución.

Orden	N	N_{nz}	p_{n}	H	C_{s}	L_{s}	K	Tiempo simulación	Tiempo modelo
IKJ	2	200	5 %	200	4	4	1	35.46	0.02
IKJ	2	200	5 %	200	4	4	2	37.60	0.01
IKJ	2	200	5 %	200	32	4	1	33.27	0.27
IKJ	2	200	5 %	200	32	4	4	38.75	0.06
IKJ	2	200	5 %	200	32	8	1	32.61	0.14
IKJ	2	200	5 %	200	256	4	2	29.15	4.33
IKJ	2	200	5 %	2000	256	16	2	338.61	1.47
IJK	2	200	5 %	200	4	4	1	43.82	0.02
IJK	2	200	5 %	200	4	4	2	47.76	0.01
IJK	2	200	5 %	200	32	4	1	43.40	0.28
IJK	2	200	5 %	200	32	4	4	51.89	0.07
IJK	2	200	5 %	200	32	8	1	42.38	0.14
IJK	2	200	5 %	200	256	4	2	40.35	4.35
IJK	2	200	5 %	2000	256	16	2	435.66	1.48
JIK	2	200	5 %	200	4	4	1	41.62	0.00
JIK	2	200	5 %	200	4	4	2	41.67	0.00
JIK	2	200	5 %	200	32	4	1	38.05	0.00
JIK	2	200	5 %	200	32	4	4	41.85	0.00
JIK	2	200	5 %	200	32	8	1	34.03	0.00
JIK	2	200	5 %	200	256	4	2	38.37	0.01
JIK	2	200	5 %	2000	256	16	2	320.59	0.01

Cuadro 3.9: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz dispersa-matriz densa.

N	N_{nz}	p_{n}	C_{s}	L_{s}	K	Tiempo dineroIII	Tiempo sim. simpl.	Tiempo modelo
2	200	5.00 %	32	8	1	6.49	0.59	0.21
2	200	5.00 %	32	8	4	6.55	0.66	0.21
10	1000	1.00 %	32	8	1	35.31	3.30	5.03
10	1000	1.00 %	32	8	4	36.61	3.87	5.17
10	1000	1.00 %	256	16	2	34.73	3.07	4.78
5	1000	4.00 %	16	8	1	35.06	3.27	1.24
5	1000	4.00 %	256	16	2	33.59 2.85		1.22
5	2000	8.00 %	16	8	1	70.06	6.55	1.25
5	2000	8.00 %	256	16	2	67.63	5.67	1.26

Cuadro 3.10: Tiempos de usuario de la simulación mediante dineroIII, el simulador simplificado y tiempos de ejecución del modelo para la trasposición de una matriz dispersa.

N	N_{nz}	p_{n}	H	B_{J}	B_{K}	C_{s}	L_{s}	K	Tiempo simulación	Tiempo modelo
4	200	1.25 %	200	20	200	32	4	1	23.19	2.20
2	100	2.50 %	200	20	200	32	4	1	8.80	1.41
2	200	5.00 %	200	20	100	32	4	1	17.28	1.41
2	200	5.00 %	200	20	200	32	4	1	14.64	1.38
2	200	5.00 %	200	20	200	32	4	2	14.75	0.66
2	200	5.00 %	200	20	200	32	8	1	14.51	0.60
2	200	5.00 %	2000	20	200	32	4	1	146.14	1.38
2	200	5.00 %	2000	20	200	128	4	1	14.38	5.73
10	1000	1.00 %	200	20	200	128	8	1	123.73	11.89
10	1000	1.00 %	400	25	400	128	8	1	176.84	11.13
10	1000	1.00 %	400	25	400	128	8	4	181.62	1.67

Cuadro 3.11: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz dispersa-matriz densa IKJ altamente optimizado.

Capítulo 4

Otras distribuciones: matrices banda

Abordaremos aquí la problemática de la adaptación de nuestra técnica de modelado analítico a distribuciones no uniformes de las entradas de las matrices dispersas. En concreto, nos centraremos en las matrices banda, muy frecuentes en problemas reales. Comenzaremos por considerar bandas uniformes [46], [23], es decir, aquellas cuyas entradas están distribuidas de forma uniforme en toda la extensión de la banda. Posteriormente extenderemos el modelado para considerar bandas no uniformes, donde distintas diagonales pueden tener distintas densidades. Hemos introducido este tipo de modelado en [18]. Hemos comprobado que un gran número de matrices pertenecientes a colecciones como la Harwell-Boeing [20] o NEP [5] obedecen a alguno de estos dos tipos de distribución.

A continuación describimos el modelado de los algoritmos utilizados en el capítulo anterior considerando que la matriz dispersa implicada es una matriz en banda. No se tratará el algoritmo descrito en la sección 3.7, dado que el desarrollo del que surgió está basado en una distribución uniforme de las entradas en la matriz, y no es el idóneo para una distribución en banda.

4.1. Bandas uniformes

A fin de simplificar las fórmulas consideraremos matrices banda cuadradas, de forma que la banda vendrá descrito por un solo valor: W, el ancho de la banda. Por otra parte, al distribuirse las entradas de cada fila a lo largo de W posiciones en lugar de N, utilizaremos como valor para la densidad

$p_{\mathrm{n}} = \beta/W$. En consecuencia, el valor de p también se ve modificado, ya que se calcula en función de p_{n} (véase el capítulo 3).

4.1.1. Producto matriz dispersa-vector

Vectores `R` y `D`

El modelado de estos vectores experimenta una pequeña modificación en el cálculo de $S_{\mathrm{int\ D}}$ en (3.6), ya que tras el procesamiento de una fila, la probabilidad uniforme de acceso no se reparte sobre los N datos del vector `X`, sino sobre tantos como caben en la banda. De ahí que el vector de área de interferencia cruzada correspondiente a los accesos a dicho vector durante el producto escalar con una fila de la matriz dispersa no sea ya $S_{\mathrm{l}}(N, p)$ sino $S_{\mathrm{l}}(W, p)$.

Vector `X`

El número de fallos sobre el vector `X` es el más afectado por la distribución en banda de las entradas. El modelado del comportamiento de este vector es idéntico al descrito en la sección 3.2.3, teniendo en cuenta los siguientes puntos:

- A la hora de calcular el vector de área de auto-interferencia, debemos considerar un tamaño de W posiciones en lugar de N por el mismo motivo. De esta forma forma, en lugar de adoptar el valor $S^i_{\mathrm{la}}(N, p)$ en (3.9), deberá ser $S^i_{\mathrm{la}}(W, p)$.
- En la ecuación (3.11) el límite del sumatorio pasa de ser M a W por estar las entradas de cada columna dispersas a lo largo de W posiciones en lugar de todas las filas de la matriz.
- El número de líneas diferentes del vector empleadas durante el producto escalar con una fila de la matriz dispersa pW/L_{s} en lugar de pN/L_{s} al distribuirse las entradas sólo sobre W de las posiciones de una fila. Este reemplazo debe realizarse en la ecuación (3.12).

Validación y análisis

La validación de este modelo se realizó con simulaciones sobre múltiples posibles combinaciones de los parámetros de entrada, mostrando la tabla 4.1

N	N_{nz}	W	p_{n}	C_{s}	L_{s}	K	σ	Δ
1	10	100	10.0 %	2	4	1	4.52	-0.43
1	10	100	10.0 %	2	4	2	0.39	0.06
1	10	100	10.0 %	4	4	4	0.01	-0.01
1	10	100	10.0 %	8	4	1	1.02	0.32
1	10	100	10.0 %	8	8	2	0.19	0.12
1	10	100	10.0 %	16	8	4	0.02	0.14
10	100	300	3.3 %	1	8	1	8.63	0.48
10	100	300	3.3 %	16	4	2	0.13	-0.01
10	100	300	3.3 %	32	8	2	0.19	0.03
100	8000	20000	0.4 %	8	8	2	0.91	1.70
100	8000	20000	0.4 %	16	8	1	4.50	3.37
100	8000	20000	0.4 %	16	8	4	0.91	3.26
100	8000	20000	0.4 %	32	8	2	4.42	3.59
100	8000	20000	0.4 %	64	16	1	1.50	0.86
100	8000	20000	0.4 %	64	8	4	0.06	0.05

Cuadro 4.1: Desviación del modelo para el producto matriz dispersa banda-vector.

algunas de ellas. En el conjunto total de simulaciones se obtuvo un error medio del modelo del 1.04 %, mientras que la desviación típica media de las simulaciones supuso un 5.71 % de la media del número de fallos obtenidos.

La relación entre W y C_{s} en el producto matriz banda dispersa-vector se muestra en la figura 4.1. En este gráfico hemos considerado bandas anchas a fin de ilustrar el efecto de las auto-interferencias en el acceso al vector `X`. La reducción del ancho de la banda tiene una gran influencia sobre el número de fallos debido a que reduce la probabilidad de auto-interferencia e incrementa la probabilidad de reuso de las líneas del vector `X`, ya que las entradas se reparten sobre filas menores (mejora de la localidad espacial) y columnas más pequeñas (mejora de la localidad temporal). Se alcanza un número de fallos próximo al mínimo cuando $C_{\text{s}} \geq 1{,}5W$, siendo de poca utilidad los incrementos de C_{s} más allá de ese valor. Es intuitivo que sólo se pueden conseguir buenas tasas de fallos con $C_{\text{s}} > W$, ya que con este valor hay una línea en la caché para cada línea de la banda de la fila que se está procesando. El espacio adicional se necesita para evitar el efecto combinado de las auto-interferencias y de las interferencias cruzadas, como demostraremos a continuación a través de un análisis de una banda fija para diversos niveles de asociatividad y tamaños de caché.

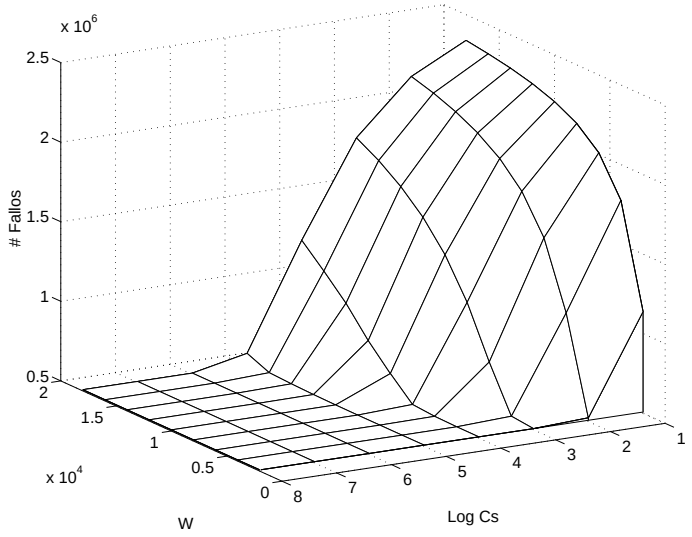

Figura 4.1: Número de fallos durante el producto matriz dispersa-vector de una matriz 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas, $L_{\mathrm{s}} = 8$ y $K = 4$ en función de W y C_{s}.

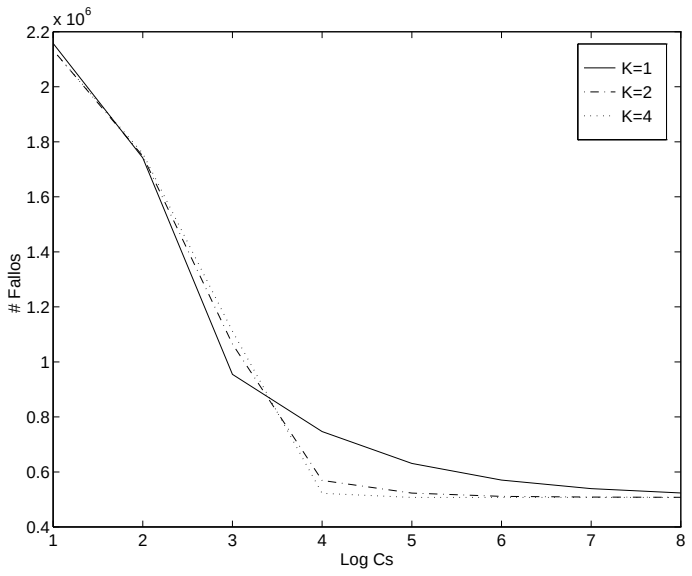

Figura 4.2: Número de fallos durante el producto matriz dispersa-vector de una matriz 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas y $W = 8000$ en función del tamaño de la caché y del grado de asociatividad para $L_{\mathrm{s}} = 8$.

En la figura 4.2 se muestra el número de fallos para una matriz con $W = 8000$ en relación al tamaño y grado de asociatividad de la caché. Podemos observar que para $K = 1$ se alcanza la mayoría de la mejora para la caché de 8K palabras debido a la eliminación de las auto-interferencias. El aumento del tamaño de la caché a partir de ese tamaño ayuda a reducir gradualmente las interferencias cruzadas. Por otro lado, las cachés con $K > 1$ presentan otro comportamiento: El gradiente en la reducción de fallos es muy grande mientras $C_s \leq$16K palabras. La razón es que en la caché de 8K palabras hay K líneas diferentes de `X` asociadas al mismo conjunto de la caché. Como las líneas se suelen acceder en el mismo orden y la caché usa un reemplazo LRU, cualquier interferencia cruzada puede provocar fallos para todas las líneas de `X` asociadas al mismo conjunto. El resultado es que las interferencias cruzadas pueden afectar a tantas líneas como pueda contener un conjunto, generándose así más fallos. De hecho podemos ver que la caché asociativa por conjuntos de cuatro vías se comporta peor que la de dos vías para este tamaño de caché.

Para las cachés con $C_s \geq$ 16K palabras, los conjuntos tienen suficientes líneas como para poder absorber los accesos a los vectores que definen la matriz dispersa y el vector destino sin incrementar las interferencias sobre `X` debido a su combinación con las líneas de este vector que residen en el mismo conjunto. Para los tamaños pequeños de caché todas las asociatividades tiene un comportamiento muy similar debido al gran número de interferencias. La conclusión es que las cachés asociativas por conjuntos ayudan a reducir el efecto de las interferencias cuando el número de líneas que compiten por un conjunto dado de la caché es menor o igual que K; de lo contrario el rendimiento es my similar al de una caché de mapeado directo. Lo que es más, en este caso, si las líneas asociadas al mismo conjunto se suelen acceder en el mismo orden, los grados de asociatividad altos pueden comportarse peor.

4.1.2. Producto matriz dispersa-matriz densa: orden JIK

El modelado de este algoritmo en el capítulo precedente se efectuó utilizando el desarrollado para el producto matriz dispersa-vector, con lo que las modificaciones requeridas para reflejar el comportamiento del vector `R` y las matrices densas `D` y `B` son análogas a las comentadas en el apartado anterior.

N	N_{nz}	W	p_{n}	H	C_{s}	L_{s}	K	σ	Δ
1	10	300	3.3 %	100	2	4	1	2.27	0.83
1	10	300	3.3 %	100	4	4	4	0.00	-0.01
1	10	300	3.3 %	100	8	4	1	1.36	0.16
1	10	300	3.3 %	100	8	8	2	0.94	-0.15
1	10	300	3.3 %	100	16	8	4	5.77	-0.87
1	10	300	3.3 %	100	64	8	2	31.95	3.07
2	20	100	10.0 %	100	1	8	1	65.67	-7.13
2	20	100	10.0 %	100	16	4	2	0.00	0.00
2	20	100	10.0 %	100	32	8	4	5.14	0.01
10	100	800	1.25 %	100	4	4	1	42.36	-9.23
10	100	800	1.25 %	100	16	4	2	0.48	0.01
10	100	800	1.25 %	100	16	8	1	3.16	-0.52
10	100	800	1.25 %	100	32	8	4	0.00	0.00
10	100	800	1.25 %	100	64	16	2	0.34	-0.06
10	100	800	1.25 %	1000	256	32	2	20.48	-1.44

Cuadro 4.2: Desviación del modelo para el producto matriz dispersa banda-matriz densa con orden JIK.

Validación

De las combinaciones de los parámetros de entrada comprobadas para este algoritmo se obtuvo una media del 3 % para el valor absoluto del error cometido por los modelos, en tanto que la desviación típica media fue del 16.98 %. La tabla 4.2 muestra los datos de validación para algunas combinaciones.

En este capítulo no analizaremos el comportamiento de estos algoritmos para matrices banda, dado que los cambios residen en que el conjunto de trabajo se limitará al tamaño de la banda W en lugar del número de filas o de columnas de la matriz dispersa, según la dimensión que se considere.

4.1.3. Producto matriz dispersa-matriz densa: orden IKJ

Se requieren cambios en el modelado de todas las estructuras de datos excepto los vectores `A` y `C`.

Vector `R`

Entre dos accesos consecutivos en el bucle más externo a este vector ya no es posible considerar que se accede a la totalidad del espacio de memoria correspondiente a la matriz `B` debido a que durante el procesamiento de cada fila de la matriz dispersa sólo son referenciables W de las filas de la matriz densa. Como las líneas correspondientes a las diferentes columnas de estas filas no ocupan posiciones consecutivas ($W < N$), tampoco podemos emplear la fórmula S_{l}. Para calcular la interferencia cruzada de esta matriz: uniremos $W/L_{\mathrm{s}}p$ veces el valor $S_{\mathrm{r}}(H, 1, N)$, es decir, una por grupo de L_{s} filas que pueda resultar referenciado durante el procesamiento de una fila de la matriz dispersa. De esta forma, la ecuación (3.19) pasaría a tener la forma:

$$S_{\text{int R}} = S_{\mathrm{s}}(\beta) \cup S_{\mathrm{s}}(\beta r) \cup S_{\mathrm{r}}(H, 1, M) \cup \bigcup^{W/L_{\mathrm{s}}p} S_{\mathrm{r}}(H, 1, N) \tag{4.1}$$

Matriz `D`

El nuevo reparto de las filas de la matriz `B` accesibles durante el procesamiento de una fila de la matriz dispersa no afecta al cálculo de los vectores de área de interferencia para la matriz `D`, puesto que siempre se trata de una sola fila (véase sección 3.4.3). No obstante, al calcular el número de fallos sí habremos de tener en cuenta que cada fila de la matriz dispersa generará aproximadamente $W/L_{\mathrm{s}}p$ accesos a líneas diferentes en cada columna de la matriz `B`, en lugar de $N/L_{\mathrm{s}}p$. De ahí que la fórmula (3.23) pase a adoptar la forma:

$$\begin{aligned} F_{\mathrm{D}} = & \frac{M}{L_{\mathrm{s}}}(1 - (1 - p_{\mathrm{n}})^{WL_{\mathrm{s}}}) \\ & (H + (\beta L_{\mathrm{s}} - Wp)S_{\text{int np D}_0} + (Wp - 1)S_{\text{int p D}_0}) \end{aligned} \tag{4.2}$$

Matriz `B`

Una vez más, el razonamiento es análogo al correspondiente al de una matriz con dispersión uniforme (sección 3.4.4) pero teniendo en cuenta que las entradas se distribuyen sobre W de las posiciones de una fila o columna. Dado que el modelado de esta matriz está totalmente derivado del correspondiente al vector `X` en el producto matriz dispersa-vector, habrá de experimentar las mismas variaciones, es decir:

- Tendremos $MW/L_{\mathrm{s}}p$ entradas, en lugar de $MN/L_{\mathrm{s}}p$, que serán la primera en su fila en generar accesos a una fila dada de `B`. De esta forma, la

ecuación (3.27) deberá expresarse como:

$$F_{\mathrm{B}} = H(pMW/L_{\mathrm{s}}(1 - P_{\mathrm{ac\ B}}) + (N_{\mathrm{nz}} - pMW/L_{\mathrm{s}})\, S_{\mathrm{int\ np\ B}_0}) \qquad (4.3)$$

- A la hora de calcular el vector de área de auto-interferencia, debemos considerar un tamaño de W posiciones en lugar de N en cada columna. La no adyacencia de estas regiones impide el uso de la fórmula $C(WH)$ para estimar el número medio de líneas que pueden interferir con otra de la matriz `B` en su conjunto. Este valor será estimado como $C((WH)/(1-S_{\mathrm{r}_K}(H,W,N)))$, de forma que los WH datos que pueden ser accedidos durante el procesamiento de una fila de la matriz dispersa se reparten entre la proporción de la caché en donde pueden recaer las líneas correspondientes a su acceso, el cual consta de regiones H de W elementos con una distancia constante N.

- Se calculará $P_{\mathrm{ac\ B}}$ como el promedio de $P_{\mathrm{ac\ B}}(j)$ para $j = 1, \ldots, W$ en lugar de $j = 1, \ldots, M$ al poderse distribuir las entradas de cada columna sólo sobre W de las filas.

Validación

A partir de las combinaciones de los parámetros de entrada comprobadas para este algoritmo se obtuvo una media del 4.28 % para el valor absoluto del error cometido por los modelos, mientras que la desviación típica media fue del 11.57 %. En la tabla 4.3 incluimos los datos de validación correspondientes a algunas combinaciones.

4.1.4. Producto matriz dispersa-matriz densa: orden IJK

Vectores `A` y `C`

La única diferencia en el modelado de estos vectores es que $S_{\mathrm{int\ J\ A}}$ debe calcularse como:

$$S_{\mathrm{int\ J\ A}} = S_{\mathrm{s}}(\beta r) \cup S_{\mathrm{s}}(1) \cup S_{\mathrm{l}}(W, p) \qquad (4.4)$$

al repartirse la probabilidad de acceso sobre una columna de la matriz `B` sobre W posiciones.

N	N_{nz}	W	p_{n}	H	C_{s}	L_{s}	K	σ	Δ
1	10	300	3.3 %	100	2	4	1	0.52	0.20
1	10	300	3.3 %	100	4	4	4	0.23	0.27
1	10	300	3.3 %	100	8	4	1	0.81	1.93
1	10	300	3.3 %	100	8	8	2	0.57	2.19
1	10	300	3.3 %	100	16	8	4	0.83	5.77
1	10	300	3.3 %	100	64	8	2	23.90	-6.01
2	20	100	10.0 %	100	1	8	1	0.27	-1.06
2	20	100	10.0 %	100	16	4	2	17.48	2.24
2	20	100	10.0 %	100	32	8	4	0.07	0.02
10	100	800	1.25 %	100	4	4	1	0.13	-0.01
10	100	800	1.25 %	100	16	4	2	0.14	0.38
10	100	800	1.25 %	100	16	8	1	0.30	0.25
10	100	800	1.25 %	100	32	8	4	0.24	0.98
10	100	800	1.25 %	100	64	16	2	0.84	-7.24
10	100	800	1.25 %	1000	256	32	2	2.24	-1.39

Cuadro 4.3: Desviación del modelo para el producto matriz dispersa banda-matriz densa con orden IKJ.

Matriz `D`

La matriz producto sólo se ve afectada en el cálculo de su vector de área de interferencia $S_{\mathrm{int\ D\ np}}$, en concreto en la porción referente a las interferencias generadas por la matriz `B`. Al restringirnos a W posiciones en cada columna de `B`, $S_{\mathrm{l}}(NH, p)$ deja de ser un buen descriptor del área de la caché afectada por los accesos a esta matriz durante el procesamiento de una fila de la matriz dispersa. Entre las diversas aproximaciones que se podrían seguir para estimar esta región hemos elegido la siguiente:

$$S_{\mathrm{B}} = \frac{2}{H} \sum_{j=1}^{H/2} \bigcup^{W/L_{\mathrm{s}}p} (S_{\mathrm{r}}(j, 1, N) \cup S_{\mathrm{r}}(H - j, 1, N)) \tag{4.5}$$

Recordemos que la porción de `B` que genera estas interferencias está formada por dos grupos de datos. Por un lado, están los asociados a las columnas con índice mayor que la que se está considerando de `D` correspondientes a las filas direccionadas por las entradas de la fila i de la matriz dispersa. Por otro lado, están los de las columnas con índice menor asociadas a las filas direccionadas por las entradas de la siguiente fila. Dado que en cada fila de la matriz dispersa las entradas se distribuyen de forma que generan accesos

a $W/L_{\mathrm{s}}p$ líneas en cada columna de la matriz densa, y son independientes en cada fila, si consideramos la columna j de la matriz `D`, tendremos que los accesos a las $W/L_{\mathrm{s}}p$ filas de líneas en las columnas con índice mayor que j tendrán un vector área $\bigcup^{W/L_{\mathrm{s}}p} S_{\mathrm{r}}(H-j,1,N)$, y los de las $W/L_{\mathrm{s}}p$ filas de líneas de las j columnas que la preceden y ella, $\bigcup^{W/L_{\mathrm{s}}p} S_{\mathrm{r}}(H-j,1,N)$. La expresión (4.5) utiliza como aproximación la media aritmética de las uniones de estos vectores.

También se ha comprobado la posibilidad consistente en seguir una aproximación igual a la seguida en la matriz `B`, es decir estimando este vector de área como:

$$S_{\mathrm{B}} = S_{\mathrm{l}}\left(\frac{WH}{1-S_{\mathrm{r}_K}(H,W,N)},p\right) \tag{4.6}$$

En este caso, para matrices con valores elevados de W y H mejora los resultados, pero los empeora en caso contrario al no tener en cuenta que la distribución de las líneas accedidas no es realmente uniforme debido a la distancia constante N que debe haber entre cada dos líneas consecutivas accedidas para una fila dada. La importancia de este factor aumenta a medida que WH disminuye y que se reduce el número de filas de la matriz dispersa contempladas en el periodo al que corresponde el cálculo del vector de área. Este es también el motivo por el que no se ha seguido esta aproximación en la estimación del vector de área de interferencia cruzada generado por la matriz `B` durante el procesamiento de una fila de la matriz dispersa con respecto al vector `R`.

Esta aproximación sin embargo sí es aplicable en el caso de la matriz `B` porque, en general, entre dos accesos consecutivos a la misma línea de la matriz `B` habrá más de una fila procesada de la matriz dispersa, y esto permitirá un reparto más uniforme de las líneas accedidas entre la superficie correspondiente a las regiones de `B` accesibles durante dichos procesamientos.

Matriz `B`

Al igual que ocurría en el orden IKJ, el modelado es idéntico al que se efectúa para la matriz con dispersión uniforme excepto en la sustitución de N por W en la expresión final de los fallos y los aspectos relacionados con el cálculo de $P_{\mathrm{ac\,B}}$ y el vector de área de auto-interferencia de este vector. Las modificaciones son las ya explicadas en el apartado correspondiente a la matriz `B` en la sección 4.1.3.

N	N_{nz}	W	p_{n}	H	C_{s}	L_{s}	K	σ	Δ
1	10	300	3.3 %	100	2	4	1	16.20	2.47
1	10	300	3.3 %	100	4	4	4	15.49	3.92
1	10	300	3.3 %	100	8	4	1	15.97	4.23
1	10	300	3.3 %	100	8	8	2	15.70	4.66
1	10	300	3.3 %	100	16	8	4	15.52	8.72
1	10	300	3.3 %	100	64	8	2	28.17	5.83
2	20	100	10.0 %	100	1	8	1	3.84	0.45
2	20	100	10.0 %	100	16	4	2	18.77	-0.39
2	20	100	10.0 %	100	32	8	4	0.01	1.16
10	100	800	1.25 %	100	4	4	1	0.05	0.37
10	100	800	1.25 %	100	16	4	2	0.14	0.44
10	100	800	1.25 %	100	16	8	1	0.20	0.48
10	100	800	1.25 %	100	32	8	4	0.29	1.14
10	100	800	1.25 %	100	64	16	2	1.22	-6.60
10	100	800	1.25 %	1000	256	32	2	0.29	-2.89

Cuadro 4.4: Desviación del modelo para el producto matriz dispersa banda-matriz densa con orden IJK.

Validación

A partir de las combinaciones de los parámetros de entrada comprobadas para este algoritmo se obtuvo una media del 5.79 % para el valor absoluto del error cometido por los modelos, mientras que la desviación típica media fue del 13.70 %. En la tabla 4.4 incluimos los datos de validación correspondientes a algunas combinaciones.

4.1.5. Trasposición de una matriz dispersa

En este algoritmo requeriremos más cambios que en los precedentes para adaptarlo al nuevo patrón de referencias debido a su mayor complejidad.

Vector `R`

La única modificación requerida para modelar el comportamiento del vector `R` durante la trasposición de una matriz banda es una nueva ecuación para

el cálculo de $S_{\text{int R}}$ (véase el apartado 3.6.2):

$$\begin{aligned} S_{\text{int R}} =& S_{\text{s}}(\beta) \cup S_{\text{s}}(\beta r) \cup S_{\text{l}}(Wr, p_r) \cup \\ & S_{gb}(W, N_{\text{nz}}/N, p_{\text{n}}) \cup S_{gb}(W, N_{\text{nz}} r/N, p_{\text{n}}) \end{aligned} \tag{4.7}$$

en donde los cambios producidos con respecto a (3.36) son:

- El cambio del vector de área de interferencia cruzada generado por el vector `RT` de $S_{\text{l}}(Nr, p_r)$ a $S_{\text{l}}(Wr, p_r)$ al distribuirse la probabilidad de acceso a este vector sobre W posiciones en cada fila.
- El vector de área de interferencia cruzada asociado a los accesos a `AT` y `CT` pasamos a expresarlo utilizando S_{gb} en lugar de S_{g} para unificar la notación con el resto del modelo de este algoritmo, ya que, como se vió en el apartado 2.3.4, $S_{gb}(b, t, P_{\text{g}}) = S_{\text{g}}(bt, t, P_{\text{g}})$. Este vector de área describe un acceso a W grupos (uno por columna con posibilidad de contener una entrada en una fila de la matriz dispersa) de N_{nz}/N elementos (la media de entradas por columna), teniendo cada grupo una probabilidad p_{n} de ser accedido, es decir, la probabilidad de que la posición correspondiente contenga una entrada. En el caso de `CT` el número de posiciones de cada grupo es $N_{\text{nz}} r/N$ al estar constituido por enteros.

Vectores `AT` y `CT`

Al igual que se hizo al explicar el modelado de la trasposición de una matriz dispersa con una distribución totalmente uniforme de las entradas, detallaremos aquí solamente la aproximación del comportamiento del vector `AT`, por tener `CT` exactamente el mismo patrón, distinguiéndose únicamente en su composición.

Si bien la fórmula general para calcular el número de fallos sobre `AT`, es la misma, debemos tener en cuenta una serie de variaciones en el cálculo de sus componentes. Así, $N_{\text{l AT}}$, el número medio de líneas de este vector referenciadas durante el procesamiento de una fila de la matriz dispersa, pasará a calcularse como:

$$N_{\text{l AT}} = \begin{cases} \frac{N_{\text{nz}}}{L_{\text{s}}} \frac{W}{N} (1 - (1 - p_{\text{n}})^{L_{\text{s}} N / N_{\text{nz}}}) & \text{si } N_{\text{nz}}/N \leq L_{\text{s}} \\ \beta & \text{si } N_{\text{nz}}/N > L_{\text{s}} \end{cases} \tag{4.8}$$

ya que si cada grupo ocupa más de una línea, habrá una posible fila accedida por cada entrada de una fila, habiendo β. Sin embargo, si el tamaño del grupo

es inferior al de la línea de la caché, como cada grupo tiene una probabilidad p_{n} de ser accedido, una línea compuesta por grupos accesibles tendrá una probabilidad media de ser accedida de $1-(1-p_{\mathrm{n}})^{L_{\mathrm{s}}N/N_{\mathrm{nz}}}$. Esta probabilidad no puede aquí aplicarse a las $N_{\mathrm{nz}}/L_{\mathrm{s}}$ líneas de que consta el vector `AT` porque en realidad sólo W de las N columnas de la matriz dispersa pueden contener elementos en una fila dada. De ahí el producto por el cociente W/N.

Por otro lado, al igual que ocurría en el modelado del comportamiento del vector `X` en el producto matriz dispersa-vector o de la matriz densa `B` en el caso del producto matriz dispersa-matriz densa, habremos de aplicar una serie de cambios en el cálculo de $P_{\mathrm{ac\ AT}}$:

- El límite del sumatorio en (3.41) pasará a ser W, al poder accederse cada grupo sólo durante el procesamiento de W filas.
- El vector de área de auto-interferencia también se modifica en consecuencia al nuevo tipo de acceso, de forma que el correspondiente a i accesos se obtendrá como $S^i_{\mathrm{gba}}(W, N_{\mathrm{nz}}/N, p_{\mathrm{n}})$.
- En el cálculo de $S_{\mathrm{cruzada\ AT}}(i)$, el vector de área de interferencia cruzada asociado al vector `RT` pasa a ser $S^i_{\mathrm{l}}(Wr, p_r)$, ya que sólo W posiciones son accesibles durante el procesamiento de cada fila de la matriz dispersa. Además el vector de área de interferencia cruzada correspondiente al vector `CT` pasará a ser $S^i_{gb}(W, N_{\mathrm{nz}}r/N, p_{\mathrm{n}})$ para amoldarse a su nuevo patrón de acceso.

Vector `C`

La única modificación que requiere el modelado de este vector es la adaptación del cálculo de los vectores de área de interferencia cruzada de los accesos a los vectores `AT` y `CT` durante el procesamiento de i filas de la matriz dispersa en el último anidamiento del algoritmo (ver figura 3.10). El cálculo de $S_{\mathrm{AT}}(i)$ y $S_{\mathrm{CT}}(i)$ pasa a ser:

$$S_{\mathrm{AT}}(i) = S^{(iL_{\mathrm{s}})/(\beta r)}_{gb}(W, N_{\mathrm{nz}}/N, p_{\mathrm{n}}) \quad (4.9)$$

$$S_{\mathrm{CT}}(i) = S^{(iL_{\mathrm{s}})/(\beta r)}_{gb}(W, N_{\mathrm{nz}}r/N, p_{\mathrm{n}}) \quad (4.10)$$

Vector `RT`

Si analizamos el comportamiento de este vector bucle a bucle, tal y como se hizo en la sección 3.6.5, veremos que el primero tiene un comportamiento

independiente de la distribución de las entradas en la matriz dispersa. En el segundo el comportamiento es idéntico al del vector `X` en el producto matriz dispersa-vector, con lo que los cambios necesarios son los mismos en la obtención de $P_{\text{ac RT}_2}$: reducción del tamaño del sumatorio sobre el que se promedian los $P_{\text{ac RT}_2}(i)$ a W y reducción del tamaño de la porción del vector que puede generar auto-interferencias en los accesos de una fila de Nr a Wr en (3.55). Lo mismo cabe decir del cálculo de $P_{\text{ac RT}_4}$, si bien en este caso también debe recordarse la modificación del cálculo del vector de área de interferencia cruzada, ya que se deberá utilizar S_{gb} en lugar de S_{g} para modelar `AT` y `CT`, ya comentado en el modelado de otros vectores. Además, al haber sólo Wr/L_{s} líneas accesibles en el procesamiento de cada fila, la expresión de F_{RT_2}, desarrollada anteriormente en (3.54) pasará a ser:

$$F_{\text{RT}_2} = pM\frac{Wr}{L_{\text{s}}}(1 - P_{\text{ac RT}_2} - P_{\text{ac ext RT}_2}) + \left(N_{\text{nz}} - pM\frac{Wr}{L_{\text{s}}}\right) S_{\text{s}_0}(1) \quad (4.11)$$

modificándose también F_{RT_4} en consecuencia. Es en la obtención de $P_{\text{ac ext RT}_2}$, y consecuentemente en $P_{\text{ac ext RT}_4}$, análoga, como se vió en la sección 3.6.5, en donde se requerirán cambios más profundos. El cálculo de $P_{\text{ac ext RT}_2}(i)$ se efectúa a partir de la fórmula:

$$P_{\text{ac ext RT}_2}(i) = \frac{\sum_{j=1}^{\text{Filas}(i,M)} (1-p_r)^{(j-1)}(1 - S_{\text{int ext RT}_{2_0}}(i, j + \text{Primera}(i) - 1))}{\text{Filas}(i, M)} \quad (4.12)$$

en donde Primera(i) da el índice de la primera fila que puede generar accesos a la i-ésima línea del vector `RT`:

$$\text{Primera}(i) = \text{máx}\{1, iL_{\text{s}}/r - W/2 + 1\} \quad (4.13)$$

y la función Filas(i, j) da el número de filas con un índice menor o igual que j que pueden generar accesos a la i-ésima línea del vector `RT`:

$$\begin{aligned}\text{Filas}(i, j) = {} & \text{mín}\{j, iL_{\text{s}}/r + W/2 + L_{\text{s}}/r - 1\} - \\ & \text{mín}\{j, \text{Primera}(i) - 1\}\end{aligned} \quad (4.14)$$

Ambas expresiones se basan en que el primer componente de la i-ésima línea de `RT` está asociado a la columna iL_{s}/r, comenzando siempre a contar las líneas desde 0. Puede comprobarse que en el caso de una matriz cuadrada la primera fila en donde dicha columna puede contener elementos es Primera(i), mientras que la última es mín$\{M, iL_{\text{s}}/r + W/2 + L_{\text{s}}/r - 1\}$. Este es también el motivo por el que al valor j del sumatorio, que tendrá un valor comprendido

entre 1 y $W + L_s/r$, se le suma Primera$(i) - 1$, para obtener el índice absoluto de la fila.

Finalmente, la expresión para el cálculo de $S_{\text{int ext RT}_2}(i,j)$ pasa a ser:

$$\begin{aligned} S_{\text{int ext RT}_2}(i,j) = & S_s(\lfloor (N_{\text{l RT}} - 1 - i)/N_k \rfloor C_{sk}) \cup \\ & \bigcup_{z=1}^{\lfloor i/N_k \rfloor} S_l(C_{sk}, 1 - (1-p_r)^{\text{Filas}(i - zN_k, j)}) \cup \\ & S_s((j - 1/2)\beta r) \end{aligned} \tag{4.15}$$

en donde puede verse que el cambio afecta sólo al cálculo del vector de área de auto-interferencia para las líneas precedentes a la que se está estudiando. En el caso de una distribución uniforme de las entradas, cualquier línea puede ser accedida durante el procesamiento de cualquier fila de la matriz dispersa, con lo que bastaba aplicar la expresión S_l, sabiendo que todas las líneas tenían una probabilidad $1 - (1 - p_r)^j$ de haber sido accedidas durante el procesamiento de las filas precedentes. En el caso de la matriz banda, sin embargo, cada una de las $\lfloor i/N_k \rfloor$ líneas que preceden a la i-ésima y que recaen en su mismo conjunto de la caché pueden tener una probabilidad de acceso distinta dependiendo del número de filas de la matriz dispersa con índice inferior o igual a j en las que fuesen accesibles, Filas$(i - zN_k, j)$.

Validación y análisis

Este modelo se validó mediante simulaciones correspondientes a cientos de posibles combinaciones de los parámetros de entrada, algunas de las cuales se muestran en la tabla 4.5. Del conjunto de simulaciones se obtuvo un error medio para nuestro modelo del 2.15 %, en tanto que la desviación típica media de las simulaciones alcanzó un 8.78 %.

En cuanto al análisis del comportamiento, las figuras 4.3 y 4.4 representan los mismos datos para este algoritmo que las figuras 4.1 y 4.2 para el producto matriz dispersa-vector, respectivamente. La primera muestra una disminución en el número de fallos con la reducción de la banda, siendo este efecto más notable en el punto donde C_s se hace mayor que W. Las razones son las explicadas para el algoritmo anterior. De todas formas, esta reducción es mucho más suave que en el producto matriz dispersa-vector porque los accesos al vector `RT` en los bucles 2 y 4, que son los más favorecidos por la reducción de la banda, representan sólo una pequeña fracción del total de fallos. Por otro lado, el número de fallos sobre los vectores `AT` y `CT`, que

N	N_{nz}	W	p_{n}	C_{s}	L_{s}	K	σ	Δ
1	10	100	10.0 %	2	4	1	12.43	-6.30
1	10	100	10.0 %	2	4	2	1.69	-5.07
1	10	100	10.0 %	4	4	4	0.74	-1.65
1	10	100	10.0 %	8	4	1	9.44	-2.61
1	10	100	10.0 %	8	8	2	8.54	-1.13
1	10	100	10.0 %	16	8	4	1.71	0.82
10	100	300	3.3 %	1	8	1	0.55	-2.02
10	100	300	3.3 %	16	4	2	2.31	-0.65
10	100	300	3.3 %	32	8	2	4.11	0.42
100	8000	20000	0.4 %	8	8	2	0.02	0.10
100	8000	20000	0.4 %	16	8	1	0.08	0.52
100	8000	20000	0.4 %	16	8	4	0.03	0.51
100	8000	20000	0.4 %	32	8	2	0.08	0.21
100	8000	20000	0.4 %	64	16	1	0.13	-0.29
100	8000	20000	0.4 %	64	8	4	0.10	0.46

Cuadro 4.5: Desviación del modelo para la trasposición de una matriz dispersa en banda.

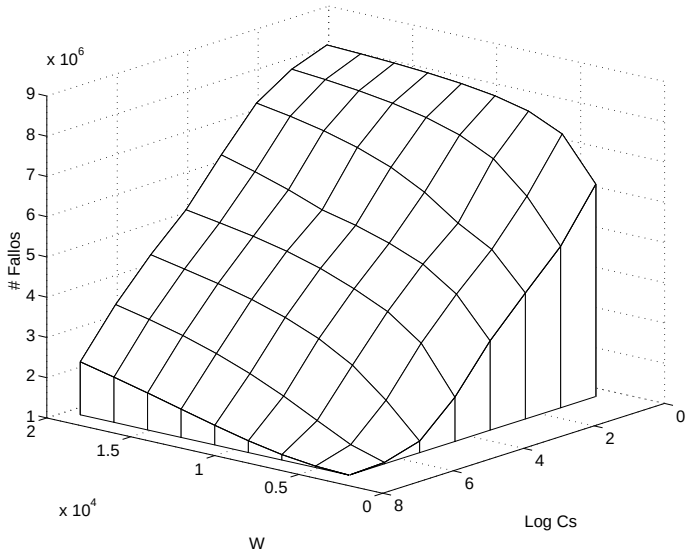

Figura 4.3: Número de fallos durante la trasposición de una matriz dispersa 20K×20K con $N_{\mathrm{nz}} = 2M$ entradas, $L_{\mathrm{s}} = 8$ y $K = 4$ en función de W y C_{s}.

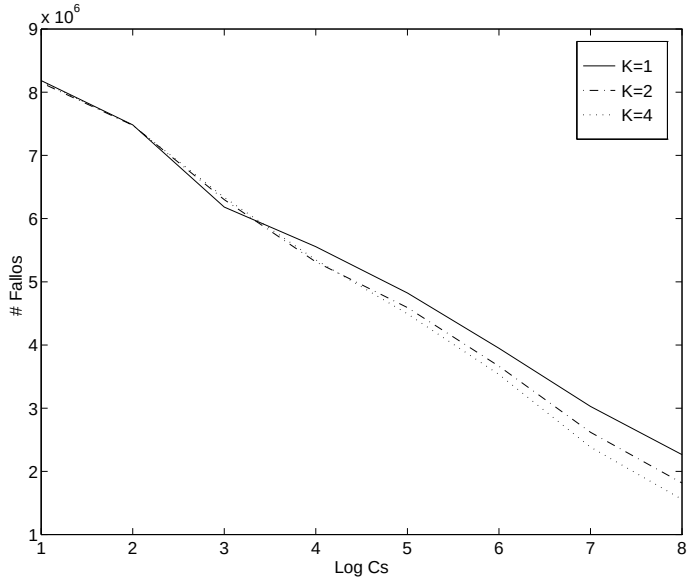

Figura 4.4: Número de fallos durante la trasposición de una matriz dispersa 20K×20K con $N_{\text{nz}} = 2M$ entradas y $W = 8000$ en función del tamaño de la caché y del grado de asociatividad para $L_{\text{s}} = 8$.

supone la mayoría del total, se reduce lentamente cuando C_{s} aumenta o W disminuye, aunque no dependen tan fuertemente del ancho de la banda. La razón es que en todos los casos mostrados en la figura los datos pertenecientes a estos vectores durante el procesamiento de todas las columnas de la banda de la matriz de entrada no cabe en la caché ($(N_{\text{nz}}/N) \cdot W$ entradas). Estos datos caben sólo cuando $W = 2000$ y $C_{\text{s}} = 256K$, y podemos ver que el número de fallos se estabiliza en esta área del gráfico. Los fallos sobre `C`, `R` y `A` permanecen prácticamente constantes debido a su acceso secuencial, obteniendo como único beneficio de la reducción del ancho de la banda una probabilidad de interferencia cruzada ligeramente menor.

La figura 4.4 muestra que el comportamiento general del algoritmo con respecto a K, el grado de asociatividad, a pesar de tener similaridades con el del producto matriz dispersa-vector, no depende únicamente de W para determinar los tamaños de caché para los que la tasa de fallos alcanza valores razonables. Como ya se ha explicado, la razón es que para $W = 8000$ ninguno de los tamaños de caché considerados contiene una porción importante de los datos que el algoritmo accede durante las W iteraciones que procesan una banda completa en el bucle cuatro, el que causa la mayoría de los fallos. Sólo se benefician del aumento de C_{s} a valores mayores que W los accesos a `RT`, principalmente en el bucle dos. Esto es especialmente notable para $K = 1$, al

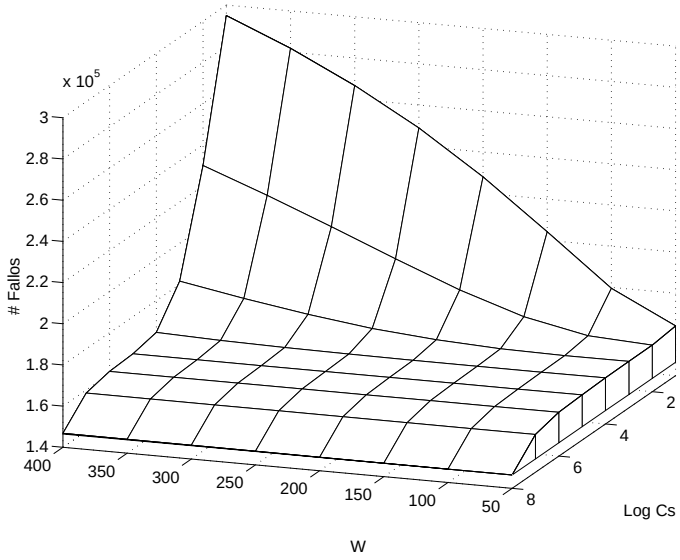

Figura 4.5: Número de fallos durante la trasposición de una matriz dispersa 5K×5K con 125K entradas, $L_s = 4$ y $K = 4$ en función de W y C_s.

igual que en el producto matriz dispersa-vector. Otra similaridad es el peor comportamiento de las cachés con $K > 1$ para un tamaño de la caché muy cercano a W debido al efecto conjugado de las interferencias cruzadas con las auto-interferencias debido a las líneas de caché que se suelen acceder en el mismo orden. Esto penaliza el algoritmo de reemplazo LRU. Las cachés asociativas rinden mejor una vez que el tamaño de la caché es notablemente mayor que el ancho de la banda, como en la figura 4.2.

A fin de averiguar los valores de los parámetros de la caché para los que el algoritmo estabiliza su número de fallos, las figuras 4.5 y 4.6 muestran los mismos datos para una matriz más pequeña usando $L_s = 4$. Durante el procesamiento de una banda de esta matriz el conjunto de trabajo para los vectores `AT` y `CT`, que contabilizan la mayoría de los fallos, es de $25W$ elementos, ya que hay una media de 25 elementos por columna. La tasa de fallos obtiene valores cercanos al mínimo en la figura 4.5 cuando el tamaño de la caché excede este valor. Los aumentos de C_s más allá de este límite mejoran poco el rendimiento. Estas mejoras sólo son perceptibles cuando el aumento del tamaño de la caché es muy grande, debido a la gran reducción de las interferencias cruzadas. Debemos tener en cuenta que este gráfico está construido para una caché de cuatro vías. Se precisarían tamaños de caché algo mayores para conseguir una buena reducción de las interferencias cruzadas en una caché de mapeado directo (ver figura 4.6).

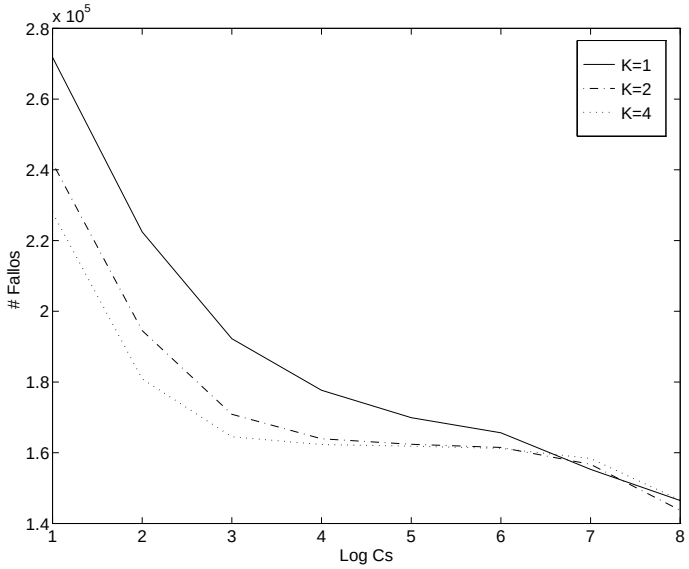

Figura 4.6: Número de fallos durante la trasposición de una matriz dispersa 5K×5K con 125K entradas y $W = 200$ en función del tamaño de la caché y del grado de asociatividad para $L_s = 4$.

Las cachés asociativas por conjuntos ayudan a reducir la tasa de fallos para cachés pequeñas en la figura 4.6 porque por término medio siempre hay menos de dos líneas compitiendo en cualquier conjunto, ya que durante el procesamiento de cada fila sólo tienen probabilidades de ser accedidas aproximadamente 200 líneas del vector `AT`, otras tantas del vector `CT` y unas pocas más pertenecientes a los restantes vectores, mientras que la menor de las cachés consideradas tiene 512 líneas. Como se esperaba, el aumento del tamaño de la caché reduce la diferencia de la tasa de fallos existente entre las cachés de mapeado directo y las asociativas por conjuntos. Para $C_s \geq 8K$ palabras los incrementos del tamaño de la caché sólo ayudan a reducir las interferencias cruzadas.

Finalmente, la relación del tamaño de la línea y de la densidad de la matriz con el número de fallos en este último algoritmo ha demostrado ser el mismo que en el producto matriz dispersa-vector, siendo la única diferencia que el gradiente del aumento del número de fallos en relación a p_n es aproximadamente tres veces mayor, lo cual era de esperar, ya que el número de accesos por cada entrada de la matriz original es ocho, mientras que en el producto matriz dispersa-vector es tres.

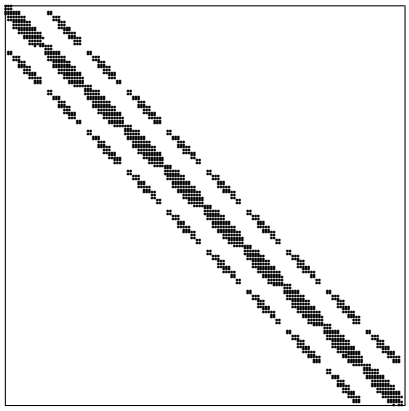

Figura 4.7: Matriz BCSSTM07, perteneciente al conjunto BCSSTRUC1 de la colección Harwell-Boeing.

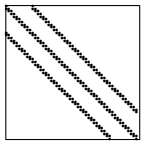

Figura 4.8: Matriz CRY10000, perteneciente al conjunto CRYSTAL de la colección NEP, y detalle de la banda.

4.2. Bandas no uniformes

Como último paso en la adaptación del modelado inicial a distribuciones no uniformes de los elementos no nulos en las matrices dispersas, hemos considerado el caso que hemos encontrado con más frecuencia en las colecciones de matrices reales. Se trata de matrices banda en las que la distribución de los elementos no nulos no es uniforme a lo largo de la banda, pero que sí puede aproximarse como uniforme en cada una de las diferentes diagonales de la banda, pudiendo tener diferentes diagonales distintas densidades. Las figuras 4.7 y 4.8 muestran matrices reales de este tipo. Así, estaremos considerando una serie de W diagonales en cada matriz con sus respectivas densidades, o lo que es lo mismo, probabilidades de que una posición perteneciente a las mismas contenga una entrada, $d_1, d_2, \ldots, d_W$.

Consideramos la probabilidad de acceso generada por los elementos residentes en un grupo de L_s columnas consecutivas. El motivo es que, como hemos visto, son las que pueden provocar un acceso a una línea del vector por el que se está multiplicando la matriz dispersa en cuestión o a una serie de H líneas de la matriz densa por la que se multiplica. Es fácil comprobar que tal conjunto de columnas puede ser afectado por accesos a lo largo del procesamiento de $W + L_s - 1$ filas de la matriz dispersa. En la primera de esas filas la probabilidad de acceso será $p_1 = d_1$, en la segunda será $p_2 = 1 - (1 - d_1)(1 - d_2)$, es decir, la opuesta a que no haya entradas ni en la primera ni en la segunda diagonal en esa fila, y así sucesivamente.

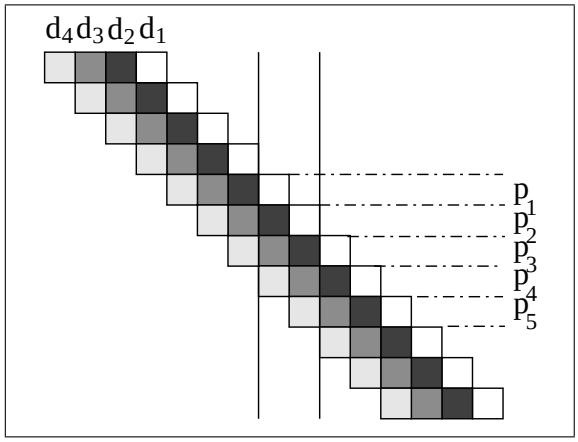

Figura 4.9: Probabilidades de que haya un no nulo en las $W + L_s - 1$ filas de la matriz que se extienden sobre $L_s = 2$ columnas consecutivas, siendo $W = 4$.

Las expresiones formales para el cálculo de las probabilidades de acceso a lo largo de las $W + L_s - 1$ filas referidas son:

$$\begin{aligned} p_i &= 1 - \prod_{j=1}^{i}(1 - d_j) \qquad i < L_s \\ p_i &= 1 - \prod_{j=i-L_s+1}^{i} (1 - d_j) \quad L_s \leq i \leq W \\ p_i &= 1 - \prod_{j=i-L_s+1}^{W} (1 - d_j) \quad W < i \end{aligned} \tag{4.16}$$

La figura 4.9 muestra un sencillo ejemplo para una banda de cuatro columnas y un tamaño de línea de dos palabras donde se pueden apreciar estas relaciones. Partiendo de estos valores extenderemos el modelo del producto matriz dispersa-vector y del producto matriz dispersa-matriz densa en sus tres ordenamientos. En las ecuaciones para las que no mencionemos cambios y que impliquen el uso de los parámetros p_n o p, éstos se seguirán derivando de la forma explicada al comienzo del apartado 4.1.

4.2.1. Producto matriz dispersa-vector

Sólo resulta afectado el modelado del vector `X`. Se adaptará la fórmula general para el cálculo de $P_{\text{ac X}}(j)$, la probabilidad de acierto en el primer acceso a una línea dada de `X` durante el producto escalar con la j-ésima fila

de la matriz dispersa por el vector:

$$P_{\text{ac X}}(j) = \sum_{i=1}^{j-1} p_i \prod_{k=i+1}^{j-1} (1-p_k)(1 - S_{\text{int X}_0}(i,j)) \tag{4.17}$$

El cálculo del vector de interferencia también variará al modificarse la distribución de las entradas. Al perderse la uniformidad, no podremos calcularlo partiendo únicamente del número de filas de la matriz dispersa procesadas desde el último acceso a la línea de `X` que estamos estudiando. De ahí que del uso de la expresión $S_{\text{int X}}(i)$ en (3.10) pasemos a $S_{\text{int X}}(i,j)$ en este caso. Su valor, que representa el vector de área de interferencia generado por los accesos que se dan durante el procesamiento de las filas comprendidas entre la i-ésima y la j-ésima de una banda dada, se calculará como:

$$S_{\text{int X}}(i,j) = S_{\text{auto X}}(i,j) \cup S_{\text{cruzada X}}(j-i) \tag{4.18}$$

El vector de área correspondiente a las interferencias cruzadas tiene la misma expresión que en el caso uniforme, sin embargo S_{I} deja de ser una expresión válida para estimar el vector de área de auto-interferencia. Para obtener el valor de este vector entre las filas i y j de una banda, debemos tener en cuenta que hay $N_{\text{D}}(j) = \lfloor \frac{j-2}{C_{\text{sk}}} \rfloor$ líneas a la derecha de la considerada que corresponden al mismo conjunto. Estas líneas tienen unas probabilidades de acceso:

$$P_{\text{D}lN_{\text{k}}}(i,j) = 1 - \prod_{k=\text{máx}\{1,i-lC_{\text{sk}}\}}^{j-lC_{\text{sk}}-1} (1-p_k), l = 1,2,\ldots,N_{\text{D}}(j) \tag{4.19}$$

Por otro lado, a su izquierda, o precediéndola en el vector, habrá otras $N_{\text{I}}(i) = \lfloor \frac{W+L_{\text{s}}-2-i}{C_{\text{sk}}} \rfloor$ con unas probabilidades respectivas de acceso:

$$P_{lN_{\text{k}}}(i,j) = 1 - \prod_{k=i+lC_{\text{sk}}+1}^{\text{mín}\{W+L_{\text{s}}-1,j-lC_{\text{sk}}\}} (1-p_k), l = 1,2,\ldots,N_{\text{I}}(i) \tag{4.20}$$

Dado que las $N_{\text{D}}(j) + N_{\text{I}}(i)$ líneas que recaen en el conjunto que estamos considerando tienen distintas probabilidades de acceso, no podemos aplicar la distribución binomial para calcular el vector de área asociado, como se hacía en el caso de la banda uniforme. La aproximación que se ha seguido ha sido la de calcular la media de líneas $\overline{L}(i,j)$ que van a ser accedidas, que es igual a la suma de las probabilidades de acceso de todas las líneas consideradas:

$$\overline{L}(i,j) = \sum_{l=1}^{N_{\text{D}}(j)} P_{\text{D}lN_{\text{k}}}(i,j) + \sum_{l=1}^{N_{\text{I}}(i)} P_{lN_{\text{k}}}(i,j) \tag{4.21}$$

A partir de este valor medio se calcula el vector de área que le corresponde, que será $S_{\text{auto X}}(i,j) = S_{\text{s}}(\overline{L}(i,j)C_{\text{sk}})$.

Finalmente, no obtendremos $P_{\text{ac X}}$ siguiendo (3.11) reemplazando N por W, como se hizo en el caso de la banda uniforme. Hay varios motivos para ello. El primero es que en este caso calcularemos la probabilidad para cada una de las $W + L_{\text{s}} - 1$ filas en las que realmente puede generarse un acceso a una línea dada. El segundo es que no se le puede asignar el mismo peso a los diferentes valores de $P_{\text{ac X}}(j)$ si sabemos que en las distintas filas hay una probabilidad diferente de que se genere un acceso. También por dichos motivos, el cálculo de F_{X} como (3.12) con los reemplazos mencionados sería inadecuado.

Por tanto se calcula como la siguiente media ponderada:

$$P_{\text{ac X}} = \frac{\sum_{j=1}^{W+L_{\text{s}}-1} p_j P_{\text{ac X}}(j)}{\sum_{j=1}^{W+L_{\text{s}}-1} p_j} \tag{4.22}$$

Los motivos antes expuestos también hacen que el cálculo de F_{X} como (3.12) usando Wp para estimar el número de accesos a una línea sea inadecuado. El cálculo deberá efectuarse como:

$$F_{\text{X}} = \sum_{j=1}^{W+L_{\text{s}}-1} p_j \frac{N}{L_{\text{s}}} (1 - P_{\text{ac X}}) + \left(N_{\text{nz}} - \sum_{j=1}^{W+L_{\text{s}}-1} p_j \frac{N}{L_{\text{s}}} \right) S_{\text{int A}_0} \tag{4.23}$$

Validación

Dado que el desarrollo de esta variedad del modelado se ha efectuado a fin de reflejar con mayor fidelidad el comportamiento de las matrices que se dan en los problemas reales, la validación de estos modelos no se ha efectuado utilizando matrices sintéticas, sino pertenecientes a las colecciones Harwell-Boeing [20] y NEP [5]. Como consecuencia, las matrices no verifican con precisión las premisas del modelo, esto es, en general, la dispersión de las entradas en cada diagonal de la banda no tiene por qué ser uniforme, además de que en muchas matrices hay grupos de elementos fuera de la banda. Por último, el tamaño de las matrices de las colecciones suele ser bastante pequeño, lo cual no favorece la convergencia de un modelo probabilístico como el nuestro.

Los datos de validación de algunas de las pruebas efectuadas se muestran

Matriz	N	N_{nz}	W	p_{n}	C_{s}	L_{s}	K	σ	Δ
bcsstk03	112	640	15	38.1 %	1	4	1	30.49	-2.40
bcsstk03	112	640	15	38.1 %	1	4	2	1.15	-0.16
bcsstk03	112	640	15	38.1 %	1	4	4	0.00	-0.12
bcsstk09	1083	18437	125	13.6 %	1	4	1	4.23	0.82
bcsstk09	1083	18437	125	13.6 %	1	4	2	0.62	0.34
bcsstk09	1083	18437	125	13.6 %	8	4	1	2.46	-0.05
bcsstk09	1083	18437	125	13.6 %	4	8	2	1.15	-0.13
bcsstm10	1086	22092	75	27.1 %	1	4	1	33.19	-3.81
bcsstm10	1086	22092	75	27.1 %	1	4	4	0.61	-0.55
bcsstm10	1086	22092	75	27.1 %	2	4	2	0.70	-0.11
bcsstm10	1086	22092	75	27.1 %	4	8	2	0.351	-0.01
cry10000	10000	49699	201	2.5 %	1	4	1	4.11	1.45
cry10000	10000	49699	201	2.5 %	1	4	4	0.68	-0.01
cry10000	10000	49699	201	2.5 %	1	8	1	53.07	-4.12
cry10000	10000	49699	201	2.5 %	16	4	2	0.22	-0.22
cry10000	10000	49699	201	2.5 %	32	8	2	0.10	-0.25
lnsp3937	3937	25407	168	3.8 %	1	4	1	8.09	-0.38
lnsp3937	3937	25407	168	3.8 %	2	8	2	2.59	-0.12
lnsp3937	3937	25407	168	3.8 %	16	16	2	0.16	-0.02

Cuadro 4.6: Desviación del modelo para el producto matriz dispersa banda no uniforme-vector.

Matriz	Orden	N_{nz}	W	p_n	H	C_s	L_s	K	σ	Δ
bcsstk21	3600	26600	251	2.9 %	100	1	4	1	0.60	1.40
bcsstk21	3600	26600	251	2.9 %	100	1	8	1	16.54	-3.11
bcsstk21	3600	26600	251	2.9 %	100	1	8	2	2.86	0.50
bcsstk21	3600	26600	251	2.9 %	100	16	4	2	0.28	0.01
bcsstk21	3600	26600	251	2.9 %	100	16	8	1	0.35	0.38
bcsstk21	3600	26600	251	2.9 %	100	32	8	2	0.01	-0.01
gr_30_30	900	7744	63	13.7 %	100	1	4	2	1.41	0.24
gr_30_30	900	7744	63	13.7 %	100	1	8	2	1.83	0.53
gr_30_30	900	7744	63	13.7 %	100	2	8	2	0.87	0.09
gr_30_30	900	7744	63	13.7 %	100	8	8	4	0.00	-0.01
gr_30_30	900	7744	63	13.7 %	100	16	4	1	35.63	2.22
gr_30_30	900	7744	63	13.7 %	100	32	8	2	24.00	8.73
cry10000	10000	49699	201	2.5 %	100	1	8	1	20.98	-4.28
cry10000	10000	49699	201	2.5 %	100	1	8	4	0.49	-0.03
cry10000	10000	49699	201	2.5 %	100	4	4	2	1.37	-0.61
cry10000	10000	49699	201	2.5 %	100	4	4	4	0.12	-0.26
cry10000	10000	49699	201	2.5 %	100	4	8	1	1.87	0.32
cry10000	10000	49699	201	2.5 %	100	4	8	2	1.38	-0.58
cry10000	10000	49699	201	2.5 %	100	32	8	1	0.28	-0.06

Cuadro 4.7: Desviación del modelo para el producto matriz dispersa banda no uniforme-matriz densa con orden JIK.

en la tabla 4.6. En este caso se obtuvo un error medio para la predicción del modelo del 1.21 % y siendo la desviación típica media del 5.09 %.

4.2.2. Producto matriz dispersa-matriz densa: orden JIK

Al igual que ocurría en el caso de las matrices con una banda uniforme, el modelado de este algoritmo requiere las mismas modificaciones que el del producto matriz dispersa-vector.

Validación

De las combinaciones de los parámetros de entrada comprobadas para este algoritmo se obtuvo una media del 1.74 % para el valor absoluto del error cometido por los modelos, en tanto que la desviación típica media fue

del 10 %. La tabla 4.7 muestra los datos de validación para algunas combinaciones.

4.2.3. Producto matriz dispersa-matriz densa: orden IKJ

Matriz `D`

El modelado de la matriz `D` explicado en la sección 4.1.3 es válido para el caso de bandas no uniformes. No obstante, en muchas matrices de las colecciones reales la mayoría de las diagonales que contienen elementos se encuentran agrupadas. El estudio de la topología de las matrices para adaptarse a esta variación tan frecuente se aleja del caso general que estamos intentando modelar, si bien puede recomendarse un modelado distinto para la matriz `D` en este caso. Se ha comprobado que en la mayoría de las matrices reales se obtiene una estimación mejor calculando el número de fallos como:

$$F_{\mathrm{D}} = \frac{MH}{L_{\mathrm{s}}} + \left(N_{\mathrm{nz}} H - \frac{MH}{L_{\mathrm{s}}} \right) S_{\mathrm{int\ D_0}} \tag{4.24}$$

en donde a los fallos intrínsecos les sumamos los debidos a las interferencias. El vector de área de interferencia se estima mediante la siguiente expresión:

$$\begin{aligned} S_{\mathrm{int\ D}} = & \frac{(\beta-1) S_{\mathrm{ra}}(H,1,M) + S'_{\mathrm{ra}}(H,1,M)}{\beta} \cup \\ & S_{\mathrm{r}}(H,1,N) \cup S_{\mathrm{s}}(1) \cup S_{\mathrm{s}}(1) \end{aligned} \tag{4.25}$$

en donde, comparando con la aproximación presentada en el apartado 3.4.3 tenemos un solo tipo de vector de área de interferencia, ya que el agrupamiento de las posiciones con entradas impide saber el número aproximado de elementos no nulos que generan referencias a las líneas de una fila de la matriz densa que han sido accedidas en la iteración previa del bucle en `K` sin un estudio más detallado de dichos grupos. Este vector utiliza siempre como vector de área de interferencia cruzada generada por `B` $S_{\mathrm{r}}(H,1,N)$ por ser el que corresponde a entradas que están cercanas y que será el que se dará con más frecuencia. También se optimiza el cálculo del vector de área de auto-interferencia, pues sabemos que de los β accesos que se generarán en cada fila de la matriz dispersa a una fila de la matriz `D`, uno (el primero) será del tipo modelado por S'_{ra}, mientras que los restantes corresponderán a las auto-interferencias que se generan al acceder exactamente a las posiciones de la fila considerada (S_{ra}).

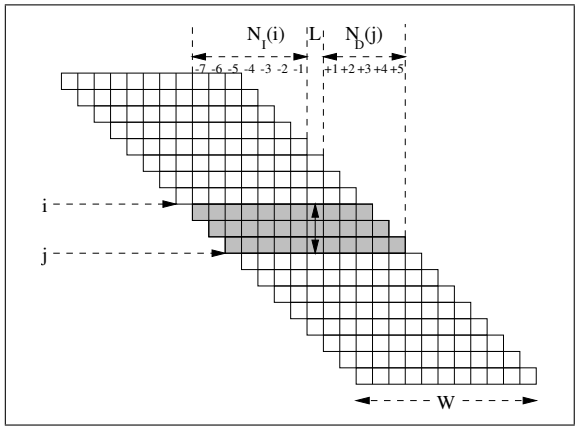

Figura 4.10: Un grupo L de L_s columnas consecutivas junto a los $N_D(j)$ grupos a su derecha y los $N_I(i)$ a su izquierda que pueden ser referenciados durante el procesamiento entre las filas i y j de la banda de L.

Matriz `B`

El modelado de los accesos a líneas de `B` que son los primeros en darse durante una iteración dada del bucle en `I` requerirán cambios de la misma naturaleza que los experimentados por el modelado del vector `X` en el producto matriz dispersa-vector. No obstante, el hecho de que `B` se acceda por filas dificultará en gran medida el cálculo del conjunto de líneas que pueden provocar interferencias con una de la fila que estamos considerando, y que es necesario ahora efectuar al no tener todas las filas la misma probabilidad de acceso. Esto es, $\overline{L}(i,j)$ deberá calcularse por separado para cada una de las líneas que conforman una fila de la matriz `B`. Una vez estimado el promedio de este valor para las H líneas de una fila, puede obtenerse el vector de área de auto-interferencia de forma análoga a la explicada en la sección 4.2.1. Al igual que ocurría allí, las interferencias cruzadas no se ven afectadas por la modificación del patrón de dispersión de los elementos no nulos en la matriz dispersa.

Para obtener $\overline{L}(i,j)$ estudiaremos las H líneas de una fila comprobando las $N_D(j) = \left\lfloor \frac{j-2}{L_s} \right\rfloor$ líneas que hay a la derecha (con índice superior) y las $N_I(i) = \left\lfloor \frac{W+L_s-2-i}{L_s} \right\rfloor$ que hay a la izquierda (precediéndola) de cada línea en cada columna de la matriz `B` para calcular cómo interactúan las líneas que constituyen las filas a las que corresponden con las de la fila considerada. La figura 4.10 ilustra esta situación, mostrando que son las únicas líneas con probabilidad de acceso durante el procesamiento comprendido entre las filas

i y j de la banda. El algoritmo seguido para efectuar esta estimación es el siguiente:

1. Numeramos las líneas de cada columna de la matriz de 0 a $\lfloor M/L_{\mathrm{s}} \rfloor$, y las de la matriz completa, considerándola por filas, de 0 a $\lfloor (MH)/L_{\mathrm{s}} \rfloor$.

2. Se elige como fila de líneas a estudiar la correspondiente a la fila central de la matriz, que comenzará por tanto en la línea $L_1 = \lfloor M/(2L_{\mathrm{s}}) \rfloor$. En pasos sucesivos se irán seleccionando las líneas de la forma $L_z = \lfloor \frac{M/2+(z-1)M}{L_{\mathrm{s}}} \rfloor, z = 1, \ldots, H$.

3. El conjunto de líneas de la matriz que recaen en el mismo conjunto puede calcularse como:

$$L_{\mathrm{CT}z} = \{u/(u - L_z) \text{ mód } N_{\mathrm{k}} = 0, 0 \leq u \leq \lfloor (MH)/L_{\mathrm{s}} \rfloor\} \tag{4.26}$$

   Ahora bien, tal como se ha dicho, sólo las $N_{\mathrm{I}}(i)$ líneas que preceden a una de la fila que estudiamos en su columna o las $N_{\mathrm{D}}(j)$ que la siguen pueden realmente generar interferencias en el intervalo de ejecución considerado. Por tanto podemos reducir más el conjunto real de líneas que pueden generar interferencias:

$$\begin{aligned} L_{\mathrm{CR}z} = \{u/u \in L_{\mathrm{CT}z}, \\ \text{máx}\{0, L_1 - N_{\mathrm{I}}(i)\} \leq \mathrm{Ind}(u) \leq \text{mín}\{\lfloor M/L_{\mathrm{s}} \rfloor, L_1 + N_{\mathrm{D}}(j)\} \end{aligned} \tag{4.27}$$

   en donde $\mathrm{Ind}(a)$ es el índice de la línea a relativo al comienzo de la columna a la que pertenece, es decir,

$$\mathrm{Ind}(a) = \left\lfloor a - \frac{N}{L_{\mathrm{s}}} \left\lfloor \frac{aL_{\mathrm{s}}}{N} \right\rfloor \right\rfloor \tag{4.28}$$

4. Las expresiones de las probabilidades de acceso para las líneas correspondientes a los grupos de L_{s} filas accesibles son (4.19) para las que se encuentran a la derecha y (4.20) para las que se hayan a la izquierda; si bien ahora l no precisa ser multiplicado por N_{k}. El motivo es que en este caso estamos tratando con líneas consecutivas, en lugar de líneas separadas por intervalos de N_{k} líneas, que era el caso del producto matriz dispersa-vector. Por este motivo, en el cálculo de los límites de los productos l no será multiplicado por C_{sk} sino por L_{s}. Así pues, ya podemos calcular $\overline{L_z}(i,j)$ para la línea que estamos estudiando como:

$$\begin{aligned} \overline{L_z}(i,j) = &\#\{u/u \in L_{\mathrm{CR}z}, \mathrm{Ind}(u) = L_1\} - 1 + \\ &\{P_{\mathrm{I}(L_1 - \mathrm{Ind}(u))}(i,j)/u \in L_{\mathrm{CR}z}, \mathrm{Ind}(u) < L_1\} + \\ &\{P_{\mathrm{D}(\mathrm{Ind}(u) - L_1)}(i,j)/u \in L_{\mathrm{CR}z}, \mathrm{Ind}(u) > L_1\} \end{aligned} \tag{4.29}$$

Matriz	Orden	N_{nz}	W	p_{n}	H	C_s	L_{s}	K	σ	Δ
bcsstk21	3600	26600	251	2.9 %	100	1	4	1	2.77	-1.97
bcsstk21	3600	26600	251	2.9 %	100	1	8	1	2.61	-0.60
bcsstk21	3600	26600	251	2.9 %	100	1	8	2	0.08	0.01
bcsstk21	3600	26600	251	2.9 %	100	16	4	2	20.32	-0.44
bcsstk21	3600	26600	251	2.9 %	100	16	8	1	74.62	7.26
bcsstk21	3600	26600	251	2.9 %	100	32	8	2	6.32	-0.26
gr_30_30	900	7744	63	13.7 %	100	1	4	2	6.00	-5.07
gr_30_30	900	7744	63	13.7 %	100	1	8	2	3.11	1.14
gr_30_30	900	7744	63	13.7 %	100	2	8	2	36.82	-4.88
gr_30_30	900	7744	63	13.7 %	100	8	8	4	32.79	-14.27
gr_30_30	900	7744	63	13.7 %	100	16	4	1	61.50	7.77
gr_30_30	900	7744	63	13.7 %	100	32	8	2	5.16	0.27
cry10000	10000	49699	201	2.5 %	100	1	8	1	5.51	-2.31
cry10000	10000	49699	201	2.5 %	100	1	8	4	0.11	0.00
cry10000	10000	49699	201	2.5 %	100	4	4	2	12.60	0.72
cry10000	10000	49699	201	2.5 %	100	4	4	4	14.02	0.11
cry10000	10000	49699	201	2.5 %	100	4	8	1	52.23	-3.11
cry10000	10000	49699	201	2.5 %	100	4	8	2	32.02	-7.21
cry10000	10000	49699	201	2.5 %	100	32	8	1	114.18	-0.89

Cuadro 4.8: Desviación del modelo para el producto matriz dispersa banda no uniforme-matriz densa con orden IKJ.

> es decir, una probabilidad de acceso uno para todas las que pertenecen a un mismo grupo de filas, por lo que se suma la cardinalidad de dicho conjunto, y la probabilidad obtenida mediante las fórmulas arriba referenciadas para las líneas de los grupos que las preceden o que las siguen y que pueden haberse accedido.

Tras repetir el proceso anterior para las H líneas de una fila, $\overline{L}(i,j)$ se estimará como la media aritmética de los valores obtenidos.

Validación

De las casi tres mil combinaciones de los parámetros de entrada comprobadas para este algoritmo se obtuvo una media del 6.87 % para el valor absoluto del error cometido por los modelos, en tanto que la desviación típica media alcanzó el 19.02 %. La tabla 4.8 muestra los datos de validación para algunas combinaciones. Puede observarse que para algunas combinaciones se

dan desviaciones importantes con respecto a la media, si bien son notoriamente inferiores a la desviación típica. El error viene dado por los redondeos cometidos al calcular el conjunto de líneas que generan interferencias con una dada, producidos por el hecho de que N, en general, no es múltiplo de L_s. Por ello las líneas con el mismo índice relativo en las diferentes columnas no contienen elementos asociados a las mismas columnas de la matriz dispersa, simplificación realizada en el modelado.

4.2.4. Producto matriz dispersa-matriz densa: orden IJK

Matriz `D`

El modelado de esta matriz se mejoró en 4.1.4 para que presentase unos resultados mejores en el caso de bandas reducidas y/o matrices con un número de columnas reducido. Este modelado se basaba en la consideración de que durante el procesamiento de una fila de la matriz dispersa se accedía aproximadamente a $W/L_\mathrm{s}p$ grupos de L_s filas de la matriz densa `B`. Dichos accesos se dividían entre los correspondientes a las columnas posteriores a la considerada de la matriz `D` en una iteración dada, y los asociados a las columnas precedentes en la iteración siguiente, que podrían corresponder, en general a grupos distintos. En el caso que nos ocupa ahora, esta uniformidad en el reparto de las entradas en una fila no es asumible.

Una forma de estimar el reparto de probabilidad del acceso a cada línea correspondiente a cada grupo de L_s posiciones en una fila de la matriz dispersa es, considerando que cada fila puede generar accesos a unas $N_{\mathrm{l\ fila}} = \lceil (W + L_\mathrm{s} - 1)/L_\mathrm{s} \rceil$ líneas, estimar la probabilidad de acceso a cada línea i como:

$$P_i = \frac{1}{L_\mathrm{s}} \sum_{j=\mathrm{máx}\{0,W-(i-1)L_\mathrm{s}-1\}}^{W-(i-2)L_\mathrm{s}-2} p_j \tag{4.30}$$

A partir de estas probabilidades podríamos estimar el número de grupos de L_s filas accedidos como $\sum_{i=1}^{N_{\mathrm{l\ fila}}} P_i$ en lugar de como $W/L_\mathrm{s}p$ en la aplicación de la expresión (4.5). Los resultados obtenidos son aceptables, si bien, en las matrices reales los fenómenos comentados en el modelado de la matriz producto en el apartado precedente pueden provocar desviaciones importantes en bandas reducidas y cuando el área de la caché afectada por los accesos a `B` durante el procesamiento de una fila de la matriz dispersa es relativamente pequeña.

Una aproximación mixta que mejora los resultados para estos casos es efectuar una simulación del acceso a H regiones de la matriz densa `B` separadas por N posiciones y constando cada una de $N_{\text{l fila}}$ líneas, es decir, el área afectada por los accesos a la matriz durante el procesamiento de una fila de la matriz dispersa. Dicha simulación usaría un único vector `CV` con una posición por conjunto de la caché (N_{k}) inicializado a ceros. El conjunto de inicio de la región i-ésima se calcula como:

$$I_i = (iN/L_{\text{s}}) \text{ mód } N_{\text{k}}, i = 0, \ldots, H-1 \tag{4.31}$$

y en cada región se suma a las posiciones $I_i, (I_i+1)$ mód $N_{\text{k}}, \ldots, (I_i+N_{\text{l fila}}-1)$ mód N_{k} del vector `CV` los valores $P_1, P_2, \ldots, P_{N_{\text{l fila}}}$ respectivamente. Al final tendremos en cada posición asociada a cada conjunto de la caché el número medio aproximado de líneas que recaerían sobre él en este acceso. Por tanto, el vector de área de interferencia cruzada asociado al acceso a `B` se estima como:

$$S_{\text{B}} = \frac{1}{N_{\text{k}}} \sum_{i=0}^{N_{\text{k}}-1} S_{\text{s}}(CV(i)C_{\text{sk}}) \tag{4.32}$$

Matriz `B`

Al igual que ocurrió en la adaptación de los modelos originales a matrices banda uniformes, los cambios requeridos en el modelado de este vector son análogos en el orden IKJ, anteriormente explicado, y el IJK.

Validación

De las combinaciones de los parámetros de entrada comprobadas para este algoritmo se obtuvo una media del 11.37 % para el valor absoluto del error cometido por los modelos, en tanto que la desviación típica media alcanzó el 11.53 %. La tabla 4.9 muestra los datos de validación para algunas combinaciones.

4.2.5. Trasposición de una matriz dispersa

El modelado de este algoritmo para matrices banda con dispersiones no uniformes lo aproximaremos empleando el modelado correspondiente a matrices banda uniformes descrito en 4.1.5 para buena parte de los cálculos, es

Matriz	Orden	N_{nz}	W	p_{n}	H	C_s	L_{s}	K	σ	Δ
bcsstk21	3600	26600	251	2.9 %	100	1	4	1	6.00	5.17
bcsstk21	3600	26600	251	2.9 %	100	1	8	1	1.26	6.56
bcsstk21	3600	26600	251	2.9 %	100	1	8	2	0.12	4.51
bcsstk21	3600	26600	251	2.9 %	100	16	4	2	10.86	6.26
bcsstk21	3600	26600	251	2.9 %	100	16	8	1	47.62	-7.68
bcsstk21	3600	26600	251	2.9 %	100	32	8	2	5.41	0.93
gr_30_30	900	7744	63	13.7 %	100	1	4	2	1.41	0.24
gr_30_30	900	7744	63	13.7 %	100	1	8	2	1.83	0.53
gr_30_30	900	7744	63	13.7 %	100	2	8	2	0.87	0.09
gr_30_30	900	7744	63	13.7 %	100	8	8	4	0.00	-0.01
gr_30_30	900	7744	63	13.7 %	100	16	4	1	35.63	2.22
gr_30_30	900	7744	63	13.7 %	100	32	8	2	24.00	8.73
cry10000	10000	49699	201	2.5 %	100	1	8	1	1.54	1.39
cry10000	10000	49699	201	2.5 %	100	1	8	4	0.00	-0.07
cry10000	10000	49699	201	2.5 %	100	4	4	2	15.03	-1.48
cry10000	10000	49699	201	2.5 %	100	4	4	4	9.49	8.71
cry10000	10000	49699	201	2.5 %	100	4	8	1	26.32	7.82
cry10000	10000	49699	201	2.5 %	100	4	8	2	39.17	-4.23
cry10000	10000	49699	201	2.5 %	100	32	8	1	53.28	7.77

Cuadro 4.9: Desviación del modelo para el producto matriz dispersa banda no uniforme-matriz densa con orden IJK.

decir, utilizando los valores medios de densidad de la banda p_{n} y de probabilidad de que haya una entrada en un grupo de L_{s} posiciones p. En particular este uso se efectuará en el cómputo de los vectores de área de interferencia cruzada, reservando el uso de expresiones que tienen en cuenta la distribución no uniforme de las entradas para el cálculo de algunos vectores de área de auto-interferencia y, especialmente, las probabilidades, de forma similar a como se hizo en el apartado 4.2.1.

Vectores `AT` y `CT`

Una vez más explicaremos aquí las modificaciones para el vector `AT`, siendo análogas las que requiere `CT`. En primer lugar, dado $G = \lceil L_{\mathrm{s}}/(N_{\mathrm{nz}}/N)\rceil$, el número de grupos de elementos asociados a una columna por línea de caché redondeado hacia arriba, calcularemos una serie de probabilidades de acceso p'_i a una línea de `AT` durante el procesamiento de $W+G-1$ filas de la matriz dispersa. Este cálculo se efectúa de forma totalmente análoga a la explicada para $p_i, i = 1, \ldots, W + L_{\mathrm{s}} - 1$ en (4.16) sustituyendo L_{s} por G.

La probabilidad de acierto en un acceso que es el primero a una línea durante el procesamiento de una fila de la matriz dispersa se calculará siguiendo una estructura semejante a la vista en la sección 4.2.1 para la del vector `X`, es decir, obtendremos la probabilidad de acierto durante el procesamiento de la j-ésima fila de una banda de la matriz dispersa mediante:

$$P_{\mathrm{ac\ AT}}(j) = \sum_{i=1}^{j-1} p'_i \prod_{k=i+1}^{j-1} (1-p'_k)(1-S_{\mathrm{int\ AT_0}}(i,j)) \tag{4.33}$$

obteniendo posteriormente $P_{\mathrm{ac\ AT}}$ como una media ponderada de $P_{\mathrm{ac\ AT}}(j)$ para $j = 1, \ldots, W+G-1$ donde el peso de cada probabilidad es precisamente la probabilidad de que en la fila correspondiente se acceda a la línea de `AT`, tal como se hizo en (4.22).

Respecto a los componentes de $S_{\mathrm{int\ AT}}$, el vector de área de interferencia cruzada lo estimaremos de forma idéntica a la explicada para el caso de una banda uniforme, mientras que para la auto-interferencia seguiremos la aproximación descrita para el vector `X` en 4.2.1. En este caso, habrá $N_{\mathrm{D}}(j) = \lfloor \frac{(j-2)(N_{\mathrm{nz}}/N)}{C_{\mathrm{sk}}} \rfloor$ líneas accesibles a la derecha y $N_{\mathrm{I}}(i) = \lfloor \frac{(W+G-2-i)(N_{\mathrm{nz}}/N)}{C_{\mathrm{sk}}} \rfloor$

líneas a la izquierda, con unas probabilidades de acceso respectivas

$$P_{\mathrm{D}lN_{\mathrm{k}}/(N_{\mathrm{nz}}/N)}(i,j) = 1 - \prod_{k=\max\{1,i-lC_{\mathrm{sk}}/(N_{\mathrm{nz}}/N)\}}^{j-lC_{\mathrm{sk}}/(N_{\mathrm{nz}}/N)-1} (1-p_k), l = 1,2,\ldots,N_{\mathrm{D}}(j) \quad (4.34)$$

$$P_{\mathrm{I}lN_{\mathrm{k}}/(N_{\mathrm{nz}}/N)}(i,j) = 1 - \prod_{k=i+lC_{\mathrm{sk}}/(N_{\mathrm{nz}}/N)+1}^{\min\{W+G-1,j-lC_{\mathrm{sk}}/(N_{\mathrm{nz}}/N)\}} (1-p_k), l = 1,2,\ldots,N_{\mathrm{I}}(i) \quad (4.35)$$

a partir de las cuales se extrae $\overline{L}(i,j)$, y de éste $S_{\text{auto AT}}$ de la misma forma que en el apartado 4.2.1.

Por último, la construcción del cálculo de la probabilidad de reuso se ha hecho de forma análoga a la explicada para el vector `X` en 4.2.1, por lo que adaptaremos también la estructura del cálculo de F_{AT} para que tenga una forma similar a (4.23):

$$F_{\mathrm{AT}} = \sum_{j=1}^{W+G-1} p'_j \frac{N_{\mathrm{nz}}}{L_{\mathrm{s}}} (1 - P_{\text{ac AT}}) + \left(N_{\mathrm{nz}} - \sum_{j=1}^{W+G-1} p'_j \frac{N_{\mathrm{nz}}}{L_{\mathrm{s}}} \right) S_{\text{int K AT}_0} \quad (4.36)$$

Vector `RT`

El modelado de este vector siempre se ha realizado siguiendo el del vector `X` en el producto matriz dispersa-matriz densa por tener el mismo patrón de acceso en los anidamientos segundo y cuarto del algoritmo de trasposición, con lo que las modificaciones que requerirá su modelado en dichos anidamientos serán idénticas a las comentadas en el apartado 4.2.1. La única salvedad residirá, como siempre, en la necesidad de considerar el tamaño r que tienen los componentes de este vector, al tratarse de enteros y en el cálculo de la probabilidad de acierto en un reuso en el primer acceso a una línea durante la ejecución del bucle, inexistente para `X`. Además, calcularemos p_{ri} para $i = 1,2,\ldots,W + L_{\mathrm{s}}/r - 1$, las probabilidades de acceder a una línea de `RT` durante cada fila de una banda. De nuevo, el mecanismo es el visto en (4.16) con el reemplazo de L_{s} por L_{s}/r.

A fin de estimar $P_{\text{ac ext RT}_2}(i)$, habremos de tener en cuenta la diferente probabilidad de acceso que puede darse para cada fila de la matriz dispersa, con lo que su expresión pasará a ser:

$$P_{\text{ac ext RT}_2}(i) = \frac{\sum_{j=1}^{\text{Filas}(i,M)} \prod_{k=1}^{j-1} (1-p_{rk})(1 - S_{\text{int ext RT}_{2_0}}(i, j + \text{Primera}(i) - 1))}{\text{Filas}(i,M)} \quad (4.37)$$

El componente de interferencia cruzada de $S_{\text{int ext RT}_2}$ se estimará de la misma forma que en el apartado 4.1.5. El de auto-interferencia está más afectado por la distribución de las entradas en la banda, pasando $S_{\text{int ext RT}_2}(i,j)$ a adoptar el valor:

$$\begin{aligned} S_{\text{int ext RT}_2}(i,j) =& S_{\text{s}}(\lfloor (N_{\text{l RT}} - 1 - i)/N_{\text{k}} \rfloor C_{\text{sk}}) \cup \\ & \bigcup_{z=1}^{\lfloor i/N_{\text{k}} \rfloor} S_{\text{l}} \left(C_{\text{sk}}, 1 - \prod_{k=1}^{\text{Filas}(i-zN_{\text{k}},j)} (1 - p_{rk}) \right) \cup \\ & S_{\text{s}}((j-1/2)\beta r) \end{aligned} \tag{4.38}$$

Validación

Los datos de validación de algunas de las pruebas efectuadas se muestran en la tabla 4.10. En este caso se obtuvo un error medio para la predicción del modelo del 3.25 %, siendo la desviación típica media del 9.21 %. En esta ocasión encontramos en la tabla algunos porcentajes de error del modelo superiores a la desviación típica asociada. El motivo es una simplificación en el modelado de `AT` y `CT`. En los modelados presentados en los apartados 3.6.3 y 4.1.5 se tenía en cuenta la posibilidad de que N_{nz}/N fuese mayor que L_{s}. En este caso una línea de estos vectores sólo experimenta accesos durante el procesado de una porción de la banda, puesto que los elementos pertenecientes a una columna de la matriz dispersa se extienden sobre varias líneas. Sin embargo, en este apartado hemos hecho un modelado más simple en el que hemos considerado que cada línea del vector `AT` tiene asociada durante todo el procesamiento de la banda $G = \lceil L_{\text{s}}/(N_{\text{nz}}/N) \rceil$ columnas, o lo que es lo mismo, grupos de elementos accesibles. La simplificación sistematiza el modelado del comportamiento del vector a partir del modelado para el vector `X` en el apartado 4.2.1, pero da lugar a una sobreestimación del número de fallos al considerar que la línea es accesible durante el procesado de toda la banda. A fin de reducir el error, en la implementación del modelo hemos considerado que si $N_{\text{nz}}/N < L_{\text{s}}$ sólo hay probabilidad real de acceso en las primeras f filas tales que $\sum_{i=1}^{f} p'_i \leq L_{\text{s}}$. Esta distribución de probabilidad será aproximadamente correcta para la primera de las líneas que contienen los elementos de una columna de la matriz dispersa, pero puede ser totalmente diferente de las que pueden tener las restantes debido a la irregularidad de la banda. De ahí que pese a lograr reducir sustancialmente el error, puedan generarse desviaciones importantes en algunos casos.

Matriz	N	N_{nz}	W	p_{n}	C_{s}	L_{s}	K	σ	Δ
bcsstk03	112	640	15	38.1 %	1	4	1	13.44	1.10
bcsstk03	112	640	15	38.1 %	1	4	2	4.18	0.46
bcsstk03	112	640	15	38.1 %	1	4	4	1.66	0.17
bcsstk09	1083	18437	125	13.6 %	1	4	1	6.73	-13.58
bcsstk09	1083	18437	125	13.6 %	1	4	2	4.38	-16.24
bcsstk09	1083	18437	125	13.6 %	8	4	1	4.97	-6.84
bcsstk09	1083	18437	125	13.6 %	4	8	2	5.74	-9.01
bcsstm10	1086	22092	75	27.1 %	1	4	1	3.44	-6.17
bcsstm10	1086	22092	75	27.1 %	1	4	4	1.74	-13.73
bcsstm10	1086	22092	75	27.1 %	2	4	2	2.10	-9.88
bcsstm10	1086	22092	75	27.1 %	4	8	2	4.84	-4.89
cry10000	10000	49699	201	2.5 %	1	4	1	6.06	6.28
cry10000	10000	49699	201	2.5 %	1	4	4	0.00	7.89
cry10000	10000	49699	201	2.5 %	1	8	1	20.32	-0.70
cry10000	10000	49699	201	2.5 %	16	4	2	1.04	0.34
cry10000	10000	49699	201	2.5 %	32	8	2	5.23	-1.91
lnsp3937	3937	25407	168	3.8 %	1	4	1	15.17	-5.40
lnsp3937	3937	25407	168	3.8 %	2	8	2	23.02	4.64
lnsp3937	3937	25407	168	3.8 %	16	16	2	9.10	-1.38

Cuadro 4.10: Desviación del modelo para la trasposición de una matriz banda no uniforme.

Capítulo 5

Una aproximación al modelado automático para patrones de acceso regulares

En los capítulos precedentes se ha desarrollado una estrategia de modelado basada en los principios explicados en el capítulo 2. Para cada algoritmo estudiado se ha construido un modelo de fallos de la caché siguiendo una estrategia común que ha sistematizado en gran medida el proceso de modelado. Consideramos que una automatización de la estimación de fallos de la caché como la que pretendemos mostrar aquí podría ser de gran interés para su integración en herramientas de optimización. Abordar esta tarea de automatización para códigos con patrones de acceso irregulares forma parte del trabajo futuro. Un posible enfoque para ello sería la utilización de las técnicas de emparejamiento de patrones de forma similar a la empleada en [31], [12] para identificar la estructura de los accesos generados por los patrones irregulares y aplicar las fómulas que hemos desarrollado en función de los patrones detectados. Sin embargo, los conceptos básicos de modelado introducidos en el capítulo 2 nos han posibilitado dar un primer paso en este sentido, asentando las bases para el tratamiento de códigos con patrones regulares. Una revisión de la bibliografía revela que incluso para patrones de acceso regulares hay muy pocos modelos analíticos cuya automatización se haya intentado [24], y su ámbito de aplicación es más restringido que el que presentamos aquí.

Analizaremos primero las construcciones más comunes en códigos densos, para después considerar posibles variantes de éstas en orden creciente de complejidad de modelado. En concreto, comenzaremos por estudiar el mod-

```
DO IZ=1, NZ, LZ
  ...
  DO I1=1, N1, L1
    DO I0=1, N0, L0
      A(fA1(IA1), fA2(IA2), ..., fAdA(IAdA))
      ...
      B(fB1(IB1), fB2(IB2), ..., fBdB(IBdB))
      ...
    END DO
    ...
    C(fC1(IC1), fC2(IC2), ..., fCdC(ICdC))
    ...
  END DO
    ...
END DO
```

Figura 5.1: Código tipo del modelado de códigos densos

elado de anidamientos perfectos con una sola referencia por estructura de datos, para luego considerar la posibilidad de que haya varias referencias por estructura. Finalmente se considerarán los anidamientos no perfectos y el reuso de datos entre bucles consecutivos.

5.1. Ámbito inicial del modelado

Consideraremos accesos a matrices de dimensiones arbitrarias en las que cada dimensión está indexada por una función afín de una de las variables que enmarcan la referencia, no pudiendo aparecer una misma variable en el indexado de más de una dimensión. Esta última restricción se hace con el objeto de modelar accesos secuenciales o con una distancia (*stride*) constante entre regiones de palabras consecutivas accedidas, que son los que hemos venido considerando como regulares. En cuanto a los bucles, tendrán un número de iteraciones predeterminado y que será el mismo en cada nueva iniciación del bucle, perteneciendo pues al tipo del bucle `DO` de FORTRAN. Finalmente, se permitirá el uso de bloques al ser una de las técnicas más comunes de optimización del acceso a memoria.

5.1.1. Anidamientos perfectos de bucles

Para explicar la automatización del modelado del comportamiento de las referencias, comenzaremos por considerar un grupo genérico de bucles en FORTRAN perfectamente anidados como los que muestra la figura 5.1. A fin de facilitar su referencia, numeramos los bucles desde cero, partiendo del más interno. En la figura vemos que los bucles tienen un tamaño predeterminado de N_i/L_i iteraciones independiente de los restantes bucles. En este lenguaje, sabemos que la referencia a un elemento de la matriz `A`:

```
A(fA1(IA1), fA2(IA2), ..., fAdA(IAdA))
```

accede a una posición de memoria que se calcula como:

$$\mathrm{Pos_A} + \sum_{x=1}^{\mathrm{d_A}} \left(f_{\mathrm{A}x}(\mathtt{I}_{\mathrm{A}x}) \prod_{j=1}^{x-1} \mathrm{d}_{\mathrm{A}j} \right) \tag{5.1}$$

donde $\mathrm{Pos_A}$ es la dirección base de la matriz `A`, $\mathrm{d_A}$ es su número de dimensiones y $\mathrm{d}_{\mathrm{A}j}$ el tamaño de la dimensión j.

Las funciones en los índices de las referencias constarán de una constante multiplicada por una de las variables de los bucles que encierran la referencia, más otra constante, es decir, serán funciones afines de la forma:

$$f_{\mathrm{A}x}(\mathrm{I}_{\mathrm{A}x}) = \Delta_{\mathrm{A}x}\mathrm{I}_{\mathrm{A}x} + K_{\mathrm{A}x},\ x = 0, 1, \ldots, \mathrm{d_A} \tag{5.2}$$

lo cual es con diferencia la situación más habitual. No obstante, en nuestra exposición obviaremos las constantes $\Delta_{\mathrm{A}x}$ que pueden multiplicar las variables en los índices, pues su tratamiento sería análogo al que daremos a los pasos, L_i, en los bucles. Su consideración exigiría simplemente tener en cuenta que en aquellos puntos donde se calcule la región afectada por los accesos de la referencia, en lugar de considerar que la distancia entre dos puntos de la dimensión x es $S_{\mathrm{A}x}$, como haremos en lo sucesivo, la distancia sería $\Delta_{\mathrm{A}x}S_{\mathrm{A}x}$.

5.1.2. Ecuaciones del número de fallos

Sea $F_i(R, p)$ el número de fallos sobre la referencia R en el nivel i con probabilidad de fallos p al acceder por primera vez a una línea de la matriz. Para calcular el número de fallos en el acceso a una matriz determinada `A` por parte de una referencia R, iremos recorriendo los bucles del más interno conteniendo la referencia a estudiar, al más externo, aplicando la siguiente regla en cada nivel:

1. Si la variable del bucle es una de las utilizadas en los índices de la referencia, pero no la correspondiente a la primera dimensión, la función que proporciona el número de fallos en la ejecución completa del bucle del nivel i en función de la probabilidad p será:

$$F_i(R,p) = \frac{N_i}{L_i} F_{i-1}(R,p) \tag{5.3}$$

   Esta aproximación se basa en la hipótesis de que la primera dimensión de cualquier matriz va a ser mayor o igual que el tamaño de la línea de la caché. Esto podría no ser cierto en el caso de cachés de segundo nivel o inferiores con un tamaño de línea grande y matrices de dimensiones reducidas, pero la conjunción de ambos factores da lugar a tasas de fallos muy pequeñas en las que el error producido es mínimo y las ventajas del análisis mediante el modelo automatizado, discutibles.

2. Si la variable del bucle no es ninguna de las empleadas en los índices de la referencia, el bucle es de reuso para la referencia que estamos estudiando. En este caso el número de fallos en el nivel se calcula como:

$$F_i(R,p) = F_{i-1}(R,p) + \left(\frac{N_i}{L_i} - 1\right) F_{i-1}(R, S_0(\mathtt{A}, i, 1)) \tag{5.4}$$

   donde $S(\mathtt{Matriz}, i, n)$ es el vector de área de interferencia que representa las líneas que pueden provocar interferencias con una cualquiera de la matriz `Matriz` tras n iteraciones del bucle del nivel i. La función (5.4) expresa el hecho de que en la primera iteración el bucle no influye en el número de fallos que se den sobre la matriz `A`, viniendo éste determinado por la probabilidad p calculada externamente. En las restantes iteraciones, sin embargo, accederemos a las mismas regiones de la matriz por la forma en la que hemos construido el código, con lo que el vector de área de interferencia en los primeros accesos a cada línea de esta región estará compuesto por el total de elementos accedidos durante una iteración de este bucle de reuso.

3. Si la variable del bucle es la utilizada en la función que indexa la primera dimensión de la matriz, en el caso de que $L_i \geq L_\mathrm{s}$ se procede como en el caso 1, ya que cada referencia se producirá a una línea diferente, como allí. En otro caso tenemos:

$$F_i(R,p) = \frac{N_i}{L_\mathrm{s}} F_{i-1}(R,p) + \left(\frac{N_i}{L_i} - \frac{N_i}{L_\mathrm{s}}\right) F_{i-1}(R, S_0(\mathtt{A}, i, 1)) \tag{5.5}$$

   es decir, la probabilidad de fallo en el primer acceso a cada línea referenciada vendrá dada por la probabilidad p calculada externamente.

Los restantes accesos se efectúan a líneas referenciadas en la iteración previa del bucle del nivel i, con lo que el vector de área de interferencia será el correspondiente a una iteración de este bucle.

En el primer nivel conteniendo la referencia se considera que $F_{i-1}(R, p)$, el número de fallos de la referencia en el nivel inmediatamente inferior, adopta el valor p. Una vez calculada la fórmula para el bucle más externo, el número de fallos se calculará como $F_z(R, 1)$, lo cual asume que no hay porciones de la matriz en la caché cuando se inicia la ejecución del código.

5.1.3. Cálculo de los vectores de área de interferencia

El cálculo de los vectores de área de interferencia $S(\texttt{Matriz}, i, n)$ se efectúa de forma automática analizando las referencias, de forma que:

- si la variable empleada en el indexado de una dimensión $\texttt{I}_h$, $h < i$, se asigna a esa dimensión un recorrido de N_h/L_h puntos con una distancia entre ellos de L_h puntos de la dimensión.
- si por el contrario $h > i$, se asigna un único punto en la dimensión.
- por último, si $h = i$, se asignan n puntos en la dimensión con una distancia L_i.

Hay una excepción a esta regla. Cuando la variable de la dimensión que se está considerando $\texttt{I}_h$ pertenece a un bucle que es de reuso para la referencia cuyo número de fallos deseamos calcular, y no hay ningún bucle que no sea de reuso para la referencia entre el del nivel i (sin incluir) y el h, sólo se considera un punto de la dimensión gobernada por $\texttt{I}_h$.

Tras un análisis de este tipo, el área de la matriz `A` afectada por los accesos generados por la referencia R durante una iteración del bucle i constaría de $N_{\mathrm{R}i1}$ regiones de una palabra en la primera dimensión, con una distancia entre cada par de regiones de $L_{\mathrm{R}i1}$ palabras. A su vez, en la segunda dimensión se habrán dado accesos en $N_{\mathrm{R}i2}$ de sus componentes, con una distancia constante entre puntos de acceso de $L_{\mathrm{R}i2}$ grupos de $\mathrm{d}_{\mathrm{A}1}$ palabras (tamaño de la primera dimensión), y así sucesivamente. De esta forma esta área se podría representar como:

$$\mathcal{R}_{\mathrm{R}i} = ((N_{\mathrm{R}i1}, L_{\mathrm{R}i1}), (N_{\mathrm{R}i2}, L_{\mathrm{R}i2}), \ldots, (N_{\mathrm{R}i\mathrm{d}_{\mathrm{A}}}, L_{\mathrm{R}i\mathrm{d}_{\mathrm{A}}})) \tag{5.6}$$

Las figuras 5.2 y 5.3 ilustran esta idea para una matriz bidimensional. El área así trazada tendrá típicamente forma de acceso secuencial o de acceso a

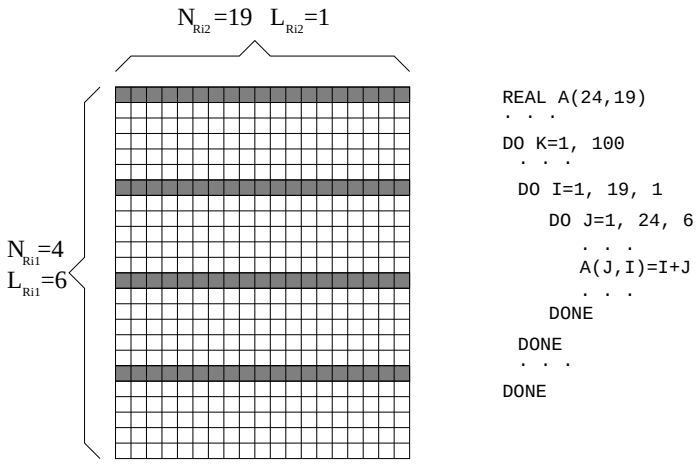

Figura 5.2: Descripción del área regular accedida de una matriz bidimensional `A` durante una iteración del bucle en `K`, en el i-ésimo nivel del anidamiento.

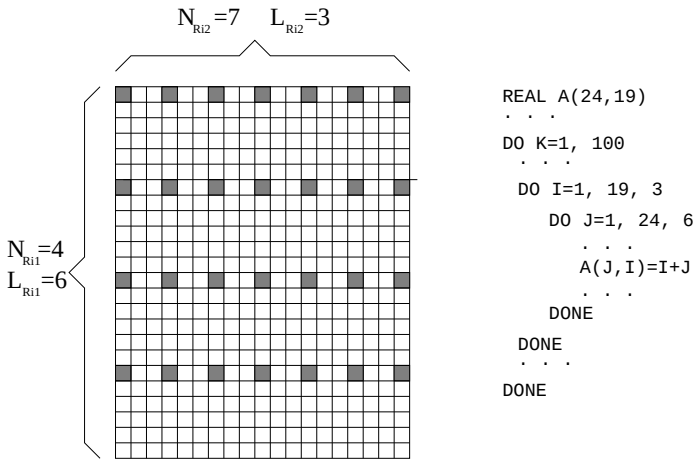

Figura 5.3: Descripción del área no regular accedida de una matriz bidimensional `A` durante una iteración del bucle en `K`, en el i-ésimo nivel del anidamiento.

varios grupos de elementos consecutivos separados por distancias constantes. Ambos accesos han sido modelados ya para el cálculo de su vector de área como auto-interferencias (referencia que se está analizando) o interferencias cruzadas (las restantes), por lo que no entraremos aquí en su modelado. Por ejemplo, el área mostrada en la figura 5.2 vendría modelada por el vector de área de interferencia $S_{\mathrm{r}}(N_{\mathrm{R}} = 76, T_{\mathrm{R}} = 1, L_{\mathrm{R}} = 6)$, cuyo cálculo se detalla en el apartado 2.3.5. En esta expresión tenemos que:

- el valor N_{R}, el número de regiones accedidas, se obtiene como el producto de los $N_{\mathrm{R}ij}$ para $j = 1, 2, \ldots, \mathrm{d_A}$,
- el tamaño T_{R} de cada región accedida es una única palabra, ya que estamos tratando una sola referencia,
- la distancia entre regiones, L_{R}, viene dada por el valor de $L_{\mathrm{R}ij}$ para el menor j tal que $N_{\mathrm{R}ij} \neq 1$ multiplicado por el tamaño de las dimensiones inferiores a j.

En el caso de que $L_{\mathrm{R}ij} = 1$, utilizariamos un modelado completamente analítico, al tratarse de un acceso secuencial a N_{R} elemenentos, que se modela mediante el vector de área de interferencia $S_{\mathrm{s}}(N_{\mathrm{R}})$ (ver apartado 2.3.1).

En el caso de que el área no correspondiese a una región regular como las comentadas (como ocurre por ejemplo en el código de la figura 5.3) se procedería al cálculo del vector de área mediante una simulación del acceso. A fin de mejorar el tiempo del modelado automático, cuando se calcula el vector de área de un determinado patrón, se lo almacena en una tabla para obtenerlo inmediatamente en caso de que haya posteriores requerimientos del analizador de modelar ese patrón.

5.1.4. Posiciones relativas de las estructuras de datos

El análisis automático de un código podría darse en circunstancias en las que se conociesen las direcciones de inicio de las estructuras de datos referenciadas en él. En este caso, a fin de aumentar la precisión del modelo, tendremos en cuenta las posiciones relativas de las matrices con respecto a la que se está estudiando al realizar la operación de unión de sus correspondientes vectores de área. Dadas dos matrices `A` y `B`, cuyos componentes pueden asociarse a T_{A} y T_{B} conjuntos de la caché, respectivamente, definimos el coeficiente de solapamiento entre ambas como:

$$\mathrm{Sol}(\mathtt{A}, \mathtt{B}) = \frac{N_{\mathrm{k}}\mathrm{Com}(\mathtt{A}, \mathtt{B})}{\mathrm{mín}\{N_{\mathrm{k}}, T_{\mathrm{A}}/L_{\mathrm{s}}\}\,\mathrm{mín}\{N_{\mathrm{k}}, T_{\mathrm{B}}/L_{\mathrm{s}}\}} \tag{5.7}$$

donde Com(A, B) es el número de conjuntos de la caché que pueden recibir líneas de ambas matrices, y cuyo calculo depende tanto de sus tamaños como de sus posiciones de inicio. Recuérdese que N_{k} es el número de conjuntos de que consta la caché. El número de conjuntos de la caché que pueden contener elementos de una matriz M se calculará como:

$$T_{\mathrm{M}} = \text{mín} \left\{ N_{\mathrm{k}}, \frac{\prod_{i=1}^{\mathrm{d_M}} \mathrm{d}_{\mathrm{M}j}}{L_{\mathrm{s}}} \right\} \tag{5.8}$$

En cuanto al cálculo de Com(A, B), partirá de la obtención del primer y del último conjunto de la caché que son referenciables para cada una de las matrices. Llammemos C_{M} y F_{M} al primer y último conjuntos, respectivamente, a los que podrá afectar la matriz M. Dada la matriz A, si $T_{\mathrm{A}} = N_{\mathrm{k}}$, entonces el primer conjunto al que podrá afectar será $C_{\mathrm{A}} = 0$ y el último será $F_{\mathrm{A}} = N_{\mathrm{k}} - 1$. En caso contrario, los obtendremos como $C_{\mathrm{A}} = \mathrm{Pos_A} \bmod N_{\mathrm{k}}$ y $F_{\mathrm{A}} = (C_{\mathrm{A}} + T_{\mathrm{A}} - 1) \bmod N_{\mathrm{k}}$. Lo mismo cabe decir para la estructura de datos B.

Una vez conocidos estos conjuntos, calculamos el número de conjuntos compartidos por ambas estructuras de datos mediante el algoritmo descrito en la figura 5.4.

A la hora de añadir un vector de área $S_{\mathtt{B}}(i)$ al vector de área de interferencia con la matriz A, se realiza el escalado:

$$\begin{array}{ll} S'_{\mathtt{B}_j}(i) & = \mathrm{Sol}(\mathtt{A},\mathtt{B}) S_{\mathtt{B}_j}(i), \quad 0 \leq j < K \\ S'_{\mathtt{B}_K}(i) & = 1 - \sum_{j=0}^{K-1} S'_{\mathtt{B}_j}(i) \end{array} \tag{5.9}$$

5.1.5. Referencias secuenciales en traslación

Por su reducido coste computacional y su eficiencia en la mejora del error, en el caso de que tanto la matriz cuyo número de fallos se está estudiando como aquella cuya probabilidad de interferencia se desea calcular tienen un acceso secuencial, estando sus referencias en traslación[1], se utiliza un procedimiento completamente diferente. Este procedimiento se basa en el cálculo del número medio de líneas l de interferencia por referencia descrito en la figura 5.5. Una vez conocido este valor, el vector de área correspondiente puede calcularse como $S_{\mathrm{s}}(lC_{\mathrm{sk}})$. El algoritmo tiene en cuenta las posiciones relativas de las matrices, con lo que no requiere el uso del coeficiente de solapamiento.

A continuación explicamos el significado de las variables P_{rst}, x_{abs} y su variable asociada x que aparecen en el citado algoritmo. Definimos $P_{\mathrm{rst}}(a_{\mathrm{abs}}, b_{\mathrm{abs}})$

[1] las referencias sólo se diferencian en las constantes sumadas a los índices de los bucles

si $F_{\mathrm{B}} \geq C_{\mathrm{B}}$ entonces
  si $F_{\mathrm{A}} \geq C_{\mathrm{A}}$ entonces
    $\mathrm{Com}(\mathtt{A}, \mathtt{B}) = \mathrm{máx}\{0, \mathrm{mín}\{F_{\mathrm{A}}, F_{\mathrm{B}}\} - \mathrm{máx}\{C_{\mathrm{A}}, C_{\mathrm{B}}\} + 1\}$
  sino
    $\mathrm{Com}(\mathtt{A}, \mathtt{B}) = \mathrm{máx}\{0, F_{\mathrm{B}} - \mathrm{máx}\{C_{\mathrm{A}}, C_{\mathrm{B}}\} + 1\} +$
    $\mathrm{máx}\{0, \mathrm{mín}\{F_{\mathrm{A}}, F_{\mathrm{B}}\} - C_{\mathrm{B}} + 1\}$
  fin si
sino
  si $F_{\mathrm{A}} \geq C_{\mathrm{A}}$ entonces
    $\mathrm{Com}(\mathtt{A}, \mathtt{B}) = \mathrm{máx}\{0, F_{\mathrm{A}} - \mathrm{máx}\{C_{\mathrm{A}}, C_{\mathrm{B}}\} + 1\} +$
    $\mathrm{máx}\{0, \mathrm{mín}\{F_{\mathrm{A}}, F_{\mathrm{B}}\} - C_{\mathrm{A}} + 1\}$
  sino
    $t = N_{\mathrm{k}} - \mathrm{máx}\{C_{\mathrm{A}}, C_{\mathrm{B}}\} + \mathrm{mín}\{F_{\mathrm{A}}, F_{\mathrm{B}}\} + 2$
    si $C_{\mathrm{B}} < F_{\mathrm{A}}$ entonces
      $\mathrm{Com}(\mathtt{A}, \mathtt{B}) = t + F_{\mathrm{A}} - C_{\mathrm{B}} + 1$
    sino
      si $F_{\mathrm{B}} > C_{\mathrm{A}}$ entonces
        $\mathrm{Com}(\mathtt{A}, \mathtt{B}) = t + F_{\mathrm{B}} - C_{\mathrm{A}} + 1$
      fin si
    fin si
  fin si
fin si

Figura 5.4: Algoritmo de estimación del número de conjuntos compartidos por dos estructuras de datos `A` y `B`.

como la probabilidad de que un acceso a una referencia que en una iteración cualquiera de los bucles accede a la posición b_{abs} provoque interferencias con un acceso a otra referencia que en esa misma iteración accede a la posición a_{abs}, siendo ambas referencias secuenciales en traslación. A su vez, definimos como capa de la caché la superficie correspondiente a la consideración de una sola línea por cada conjunto de la caché, es decir, un área de $C_{\text{sk}} = N_{\text{k}} L_{\text{s}}$ posiciones. En primer lugar, pasaremos de posiciones de memoria a posiciones en una capa de la caché obteniendo $a = a_{\text{abs}}$ mód C_{sk} y $b = b_{\text{abs}}$ mód C_{sk}. Debido a la naturaleza circular de los accesos secuenciales a la caché (tras acceder el conjunto $N_{\text{k}} - 1$ se accede al 0), para poder calcular de forma directa la distancia entre los dos puntos de acceso, si $a < L_{\text{s}}$ y $b > C_{\text{sk}} - L_{\text{s}}$, a se incrementará en C_{sk} unidades; y recíprocamente, si $b < L_{\text{s}}$ y $a > C_{\text{sk}} - L_{\text{s}}$, b será incrementado en C_{sk} unidades. La probabilidad, $P_{\text{rst}}(a_{\text{abs}}, b_{\text{abs}})$, de que los accesos a la referencia que afecta a b_{abs} (b en la caché) interfieran con la de a_{abs} (a en la caché), se calcula como:

$$P_{\text{rst}}(a_{\text{abs}}, b_{\text{abs}}) = \begin{cases} \frac{L_{\text{s}} - (b-a)}{L_{\text{s}} - 1} & \text{si } b > a \text{ y } (b-a) < L_{\text{s}} \\ \frac{L_{\text{s}} - 1 - (a-b)}{L_{\text{s}} - 1} & \text{si } a \geq b \text{ y } (a-b) < L_{\text{s}} - 1 \\ 0 & \text{en otro caso} \end{cases} \tag{5.10}$$

El algoritmo descrito en la figura 5.5 calcula el promedio de líneas l que la referencia secuencial que accede al intervalo de posiciones consecutivas $[c_{\text{abs}}, f_{\text{abs}}]$ en el bucle considerado aporta a la interferencia con la referencia que comienza su intervalo correspondiente de acceso en la posición q_{abs}.

5.1.6. Blocking

El anidamiento que hemos estudiado hasta ahora no considera construcciones como la mostrada en la figura 5.6. Esta construcción se encuentra de forma típica cuando aplicamos una técnica de transformación de bucles tan extendida como el *blocking*.

En lo referente al cálculo de los vectores de área $S(\texttt{Matriz}, i, n)$, cuando se considera la dimensión de una matriz sobre la que se ha efectuado un *blocking* (indexada por $\texttt{I}_j$ en este ejemplo), caben las siguientes posibilidades:

1. Si $i \leq j$, sólo se accede a un punto en esa dimensión.
2. Si $j < i < h$, se accede a L_h / L_j puntos con distancia L_j.
3. Si $i = h$, se accede a nL_h / L_j puntos con una distancia L_j.

$q = q_{\text{abs}}$ mód C_{sk}
$c = c_{\text{abs}}$ mód C_{sk}
$f = f_{\text{abs}}$ mód C_{sk}
$d = (f - c + 1)$ mód C_{sk}
$l = \lfloor (f - c)/C_{\text{sk}} \rfloor$
si $c \leq f$ entonces
  si ($q \geq c$ o $q = (c - 1)$ mód C_{sk}) y $q \leq f$ entonces
    $l = l + 1$
    si ($d = 0$) entonces
      $l = l + P_{\text{rst}}(q, f)$
    fin si
  sino
    $l = l + \text{máx}\{P_{\text{rst}}(q, c), P_{\text{rst}}(q, f)\}$
  fin si
sino
  si $q \geq (c - 1)$ entonces
    $l = l + 1 + P_{\text{rst}}(q, f)$
  sino
    si $q \leq f$ entonces
      $l = l + 1 + P_{\text{rst}}(q, c)$
    sino
      $l = l + P_{\text{rst}}(q, c) + P_{\text{rst}}(q, f)$
    fin si
  fin si
fin si

Figura 5.5: Algoritmo de estimación del número de líneas que una referencia secuencial aporta a otra.

```
DO Ih=1, Nh, Sh
  ...
  DO Ij=Ih, Ih+Sh-1, Sj
     ...
  END DO
    ...
END DO
```

Figura 5.6: Construcción típica de bucles con *blocking*

4. Si $h < i$, se accede a N_h/L_j puntos con una distancia L_j.

teniendo también en cuenta la excepción de cuando el bucle es de reuso para la matriz cuyo número de fallos se va a calcular.

El uso del *blocking* también debe tenerse en cuenta a la hora de calcular el coeficiente de solapamiento entre dos matrices durante la ejecución de un bucle dado, puesto que la matriz sobre la que se efectúa el *blocking* no es accedida en la totalidad de su área. En la práctica en estos casos hemos estimado el coeficiente de solapamiento como la media aritmética de los coeficientes de solapamiento que se pueden calcular para cada uno de los bloques en que se pueden descomponer las dos matrices implicadas en el cálculo.

En cuanto al cálculo de la ecuación del número de fallos para la referencia R en cada iteración de un bucle como el h, que gobierna un *blocking*, se obtiene como:

$$F_h(R,p) = \frac{N_h}{L_h} F_{h-1}(R,p) \tag{5.11}$$

puesto que iteraciones diferentes referenciarán áreas diferentes de la matriz.

5.2. Referencias múltiples a una estructura de datos

Una vez contemplado el comportamiento de una única referencia a una estructura de datos dada en un anidamiento de bucles, automatizaremos el análisis para algoritmos densos en los que hay varias referencias a algunas de las matrices o vectores. Estos accesos se diferencian típicamente en el uso de una constante sumada en una de las dimensiones. En un anidamiento perfecto estarán localizados en el mismo nivel. Aquí consideraremos este caso restringiéndonos en primer lugar al más habitual, aquél en el que la constante aparece siempre en la misma dimensión, es decir, referencias del tipo `A(..., Ii+Ki1, ...)`, `A(..., Ii+Ki2, ...)` manteniéndose iguales las expresiones de las restantes dimensiones. Posteriormente veremos cómo se modelan casos en los que hay diferencias en varias dimensiones.

5.2.1. Formulación del número de fallos en la dimensión donde difieren las referencias

En este caso ordenaremos las $\Re$ referencias en orden decreciente del valor de la constante K_{ij} ($K_{i1} > K_{i2} > \cdots > K_{i\Re}$) y modelaremos la primera

de la forma descrita hasta el momento. Para cada de una de las siguientes referencias modelaremos de la misma forma todos los bucles excepto el correspondiente a $\mathtt{I}_i$. En este caso aplicaremos un algoritmo, que depende de la diferencia entre el valor de la constante para la referencia analizada anteriormente y la actual ($\delta = K_{i(j-1)} - K_{ij}$).

Las referencias difieren en el indexado de la primera dimensión

Sea $\epsilon = \text{mcd}(L_\text{s}, L_i)$, el acceso a lo largo de este bucle puede considerarse como $G = (N_i\epsilon)/(L_i L_\text{s})$ grupos de L_i/ϵ líneas, en donde cada referencia accede L_s/ϵ posiciones diferentes. Estas posiciones pueden clasificarse de la siguiente forma:

- Hay $N_\text{misma} = \text{máx}\{0, \frac{L_\text{s}-\delta-p+\epsilon-1}{\epsilon}\}$ posiciones donde en la misma iteración tanto la referencia actual como la analizada anteriormente acceden a la misma línea, siendo p la dirección del primer acceso módulo ϵ, es decir, la menor de las posiciones relativas en una línea de la caché que corresponde a un acceso generado por la referencia que estamos estudiando. Para estos accesos la probabilidad de fallo depende sólo de los accesos entre ambas referencias en la misma iteración.

  En la notación introducida hasta este momento en este apartado para referirnos al cálculo de los vectores de área de interferencia sólo se habían considerado iteraciones completas de bucles de cierto nivel. Aquí es preciso considerar dos referencias R_1 y R_2, situadas en un bucle de nivel i, y calcular el vector de área de interferencia $S'(R_1, R_2)$ asociado a las referencias que pueda haber entre ambas durante una iteración de dicho bucle.

- Tenemos $N_\text{propia} = \text{máx}\{0, \frac{L_\text{s}-L_i-p-\text{máx}\{N_\text{misma}-1,0\}\epsilon+\epsilon-1}{\epsilon}\}$ posiciones en las que la referencia analizada accede a una línea que ha sido traída a la caché en la iteración anterior, pero que no ha sido accedida durante la iteración actual por la referencia que la precede (la que tiene la constante inmediatamente mayor). En ellas la probabilidad de fallo corresponde al primer componente del vector de área de interferencia correspondiente a una iteración completa del bucle estudiado.

- En las restantes $L_\text{s}/\epsilon - N_\text{misma} - N_\text{propia}$ posiciones la linea accedida ha sido accedida por la referencia anterior hace $\delta/\,\text{máx}\{L_i, L_\text{s}\}$ iteraciones del bucle.

Por otra parte, hay $\delta/\max\{L_i, L_s\}$ posiciones que nunca son accedidas por la referencia anterior, con lo que el número de fallos sobre cada una de ellas puede estimarse como $F_{i-1}(R_j, p)$.

Así pues, la fórmula para el número de fallos generados por la referencia R_j en el bucle i que indexa la primera dimensión y que es en la que se dan las distintas constantes que distinguen las referencias sobre la estructura de datos a la que accede R_j es:

$$\begin{aligned} F_i(R_j, p) =& N_{\text{misma}} S'(R_{j-1}, R_j) + N_{\text{propia}} S_0(\mathtt{A}, i, 1) + \\ & \left(\frac{L_s}{\epsilon} - N_{\text{misma}} - N_{\text{propia}}\right) S_0\left(\mathtt{A}, i, \frac{\delta}{\max\{L_i, L_s\}}\right) + \\ & \frac{\delta}{\max\{L_i, L_s\}} F_{i-1}(R_j, p) \end{aligned} \tag{5.12}$$

Las referencias difieren en el indexado de otra dimensión

En este caso caben dos posibilidades:

1. Si $\delta \bmod L_i = 0$, la referencia accederá a la misma línea que la referencia precedente en el ordenamiento exactamente δ/L_i iteraciones después del bucle. Por tanto la fórmula del número de fallos será:

$$F_i(R_j, p) = \frac{N_i - \delta}{L_i} F_{i-1}(R_j, S_0(\mathtt{A}, i, \delta/L_i)) + \frac{\delta}{L_i} F_{i-1}(R_j, p) \tag{5.13}$$

2. En caso contrario nunca accederán a la misma línea, con lo que la fórmula se expresa como:

$$F_i(R_j, p) = \frac{N_i}{L_i} F_{i-1}(R_j, p) \tag{5.14}$$

Por otra parte, a fin de reducir las tasas de error en el caso de que la primera dimensión de la matriz, d_{A1}, sea menor que el tamaño de la línea, L_s, cuando la distancia entre las posiciones accedidas por las referencias, $\delta \mathrm{d}_{\mathrm{A1}}$, es también menor que L_s, la fórmula del cálculo del número de fallos sobre el bucle k que gobierna la primera dimensión se modifica así:

$$F_k(R_j, p) = \frac{N_j}{L_s} F_{k-1}(R_j, p) + \left(\frac{N_j}{L_j} - \frac{N_j}{L_s}\right) \frac{\delta \mathrm{d}_{\mathrm{A1}}}{L_s} F_{k-1}(R_j, S_0(\mathtt{A}, j, 1)) \tag{5.15}$$

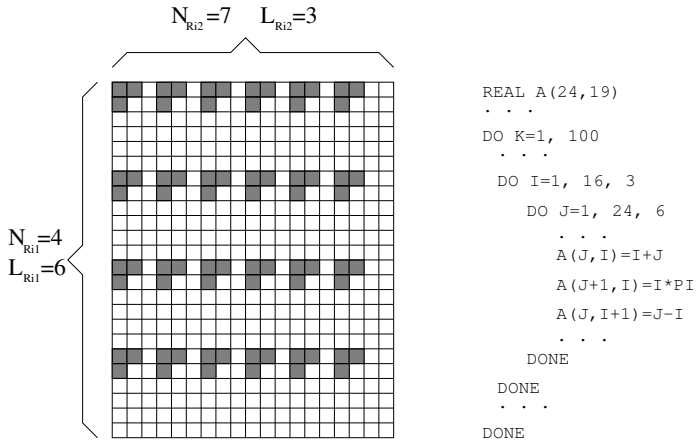

Figura 5.7: Porciones de una matriz bidimensional `A` con múltiples referencias que han resultado accedidas durante una iteración del bucle en `K`.

5.2.2. Referencias que difieren en varias dimensiones

Si hay varias dimensiones con diferencias en las constantes sumadas a las variables que las gobiernan, las referencias se ordenan en orden decreciente de la posición a la que acceden, como se hacía en la sección 5.2.1. También de la misma forma, la primera referencia se procesa como si fuese la única. Las posteriores se procesan aplicando las fórmulas vistas en los apartados anteriores y considerando como única dimensión que varía con respecto a la referencia precedente en el ordenamiento la mayor de las que difieren. Esta estrategia simplifica el tratamiento del problema permitiendo buenas aproximaciones cuando las variaciones introducidas por las diferencias de las constantes en las dimensiones menores son mucho menores que la que produce la primera dimensión con diferencias, lo cual, por otra parte, es lo más habitual.

5.2.3. Cálculo de vectores de área

La existencia de múltiples referencias a una estructura de datos hace más compleja la representación del área a la que dichas referencias pueden afectar, pues el patrón es mucho menos regular, como muestra la figura 5.7. Afortunadamente patrones como los de la figura no son los más habituales. Dado que en la gran mayoría de los códigos los accesos regulares o bien son puramente secuenciales o bien constan de regiones accedidas separadas por distancias constantes, extenderemos la notación introducida en el apartado 5.1.3 para

incluir un nuevo parámetro: el número $T_{\mathrm{R}i}$ de palabras consecutivas accedidas en la primera dimensión de que consta la región correspondiente a la referencia R en el nivel de anidamiento i. No consideramos aquí la posibilidad de accesos a puntos consecutivos en otras dimensiones separados por otros puntos no accedidos, que daría lugar a un parámetro de este tipo para cada dimensión, porque este acceso no correspondería a ninguno de los modelados en este trabajo. No obstante, esta posibilidad podría considerarse en implementaciones más genéricas.

No obstante, el analizador automático obtiene la representación de la totalidad del área afectada por los accesos generados por las diferentes referencias a la matriz a partir precisamente de la adición de las regiones afectadas por los accesos de cada referencia. En este proceso se debe tener en cuenta el solapamiento que se puede dar entre ellas en las dimensiones donde las referencias pueden dar lugar a reuso de línea de caché. Por ello una vez que se calculan las regiones de una matriz afectadas por todas las referencias a la misma, el analizador habrá de compararlas intentando fusionarlas para no contar como fuente de interferencia varias veces aquellas líneas que se vean afectadas por más de una referencia en la misma matriz. Si se verifican las condiciones que hemos impuesto en las diferencias entre las referencias, que por otra parte son las más típicas, el analizador será capaz de fusionarlas en una forma regular como la descrita.

Para facilitar el proceso de fusión de las regiones, además de los parámetros introducidos en la sección precedente para describir la región afectada por una referencia `R` en el nivel i del anidamiento, $\mathcal{R}_{\mathrm{R}i}$, el área asociada se identificará mediante la posición de la primera palabra Q_{R} que contiene, calculada aplicando (5.1) para el menor valor de cada una de las variables que se utilizan en el indexado de las dimensiones de la matriz en la referencia estudiada. En el proceso usaremos la terna

$$\mathcal{R}_{\mathrm{ext}\ ri} = (Q_r, T_{ri}, \mathcal{R}_{ri}) \tag{5.16}$$

para representar la región obtenida tras la unión de las áreas asociadas a las referencias 1 a r, ambas incluidas. Inicialmente haremos $\mathcal{R}_{\mathrm{ext}\ 1i} = (Q_{\mathrm{R}_1}, 1, \mathcal{R}_{\mathrm{R}_1 i})$. Para $r = 2, \ldots, \Re$, donde $\Re$ es el número total de referencias sobre la matriz estudiada, se aplicará el siguiente algoritmo:

1. Si $Q_r \leq Q_{\mathrm{R}_r} \leq Q_r + T_{ri} + L_{\mathrm{s}} - 2$, $T_{(r+1)i} = \mathrm{máx}\{T_{ri}, Q_{\mathrm{R}_r} - Q_r + 1\}$.

2. Si $Q_r - L_{\mathrm{s}} + 1 \leq Q_{\mathrm{R}_r} \leq Q_r + T_{ri} - 1$, $Q_{r+1} = \mathrm{mín}\{Q_r, Q_{\mathrm{R}_r}\}$ y $T_{(r+1)i} = Q_r + T_{ri} - Q_{r+1}$.

3. Para obtener un patrón regular es necesario que sólo haya distancias entre regiones accedidas en una de las dimensiones de la matriz. Así pues, en el caso de que las áreas no sean fusionables en los accesos asociados a la primera dimensión de la matriz, que son los casos contemplados en los dos puntos anteriores, detectaremos la dimensión en la que se da el salto como:

$$\mathrm{D_{salto}} = \mathrm{mín}\left\{ j, 1 \leq j \leq \mathrm{d_A} / \frac{|Q_{\mathrm{R}_r} - Q_r|}{T_{\mathrm{ac}}(\mathrm{A}, j)} \leq \mathrm{d_{A}}j \right\} \tag{5.17}$$

donde $T_{\mathrm{ac}}(\mathrm{A}, j)$ es el tamaño acumulativo de las dimensiones inferiores a la j para la matriz `A`, es decir,

$$T_{\mathrm{ac}}(\mathrm{A}, j) = \prod_{k=1}^{j-1} \mathrm{d_A}k \tag{5.18}$$

En el caso de que $N_{\mathrm{R}_r i \mathrm{D_{salto}}} = \mathrm{d_{AD_{salto}}}$ no es necesario modificar el número de puntos de que consta la región en la dimensión del salto, puesto que se la recorre por completo. No obstante, dado que se permiten variaciones en el tamaño y punto de inicio de la región accedida en la primera dimensión, deben fusionarse las regiones recalculando el punto de inicio del área global como:

$$\begin{aligned} Q_{r+1} = & \mathrm{Pos_A} + \left\lfloor \frac{\mathrm{mín}\{Q_r, Q_{\mathrm{R}_r}\}}{T_{\mathrm{ac}}(\mathrm{A}, \mathrm{D_{salto}})} \right\rfloor T_{\mathrm{ac}}(\mathrm{A}, \mathrm{D_{salto}}) + \\ & \mathrm{mín}\{(Q_r - \mathrm{Pos_A}) \bmod \mathrm{d_{A1}}, (Q_{\mathrm{R}_r} - \mathrm{Pos_A}) \bmod \mathrm{d_{A1}}\} \end{aligned} \tag{5.19}$$

y actualizando consecuentemente el tamaño de la región accedida en la primera dimensión como:

$$\begin{aligned} T_{(r+1)i} = \mathrm{máx}\{ & (Q_r + T_{ri} - 1 - \mathrm{Pos_A}) \bmod \mathrm{d_{A1}}, \\ & (Q_{\mathrm{R}_r} - \mathrm{Pos_A}) \bmod \mathrm{d_{A1}}\} - Q_{r+1} \end{aligned} \tag{5.20}$$

En otro caso, consideraremos la diferencia en puntos de la dimensión entre los puntos de inicio de la región acumulada y la correspondiente a la referencia estudiada:

$$Q_{\mathrm{dist}} = \left| \left\lfloor \frac{Q_r - \mathrm{Pos_A}}{T_{\mathrm{ac}}(\mathrm{A}, \mathrm{D_{salto}})} \right\rfloor - \left\lfloor \frac{Q_{\mathrm{R}_r} - \mathrm{Pos_A}}{T_{\mathrm{ac}}(\mathrm{A}, \mathrm{D_{salto}})} \right\rfloor \right| \tag{5.21}$$

Si $Q_{\mathrm{dist}} \bmod L_{\mathrm{R}_r i \mathrm{D_{salto}}} \neq 0$ no tendríamos, en general, un acceso a regiones separadas por una distancia constante, situación que como

hemos mencionado, no consideraremos aquí por requerir una simulación para calcular su vector de área asociado y por su infrecuencia. Únicamente en el caso de que hubiese $L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}}$ referencias con constantes de desplazamiento $0, 1, \ldots, L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}} - 1$ podríamos unificarlas en un acceso a regiones separadas por una distancia constante, ya que se accedería a $N_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}} L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}}$ puntos consecutivos en la dimensión. En este caso calcularíamos el nuevo valor para $Q_(r+1)$ y $T_{(r+1)i}$ de la forma explicada arriba y modificaríamos $N_{\mathrm{R}i\mathrm{D}_{\mathrm{salto}\ (r+1)}}$ y $L_{\mathrm{R}i\mathrm{D}_{\mathrm{salto}\ (r+1)}}$ para hacerlos adoptar los valores $N_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}} L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}}$ y 1, respectivamente.

Por otro lado, si Q_{dist} mód $L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}} = 0$, la distancia entre las dos regiones se recorre en $Q_{\mathrm{dist}}/L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}}$ iteraciones del bucle que controla la dimensión donde se da el salto. Sea

$$N_{\mathrm{R\ min}i\mathrm{D}_{\mathrm{salto}}} = \begin{cases} N_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}} & \text{si } Q_{\mathrm{R}_r} < Q_r \\ N_{\mathrm{R}i\mathrm{D}_{\mathrm{salto}\ r}} & \text{en caso contrario} \end{cases} \tag{5.22}$$

si $Q_{\mathrm{dist}}/L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}} > N_{\mathrm{R\ min}i\mathrm{D}_{\mathrm{salto}}}$, las regiones no llegan a alcanzarse, con lo que no debe realizarse ningún tipo de fusión. Es el único caso en el que podríamos tener varias regiones para una estructura de datos. En caso contrario recalculamos Q_r y T_{ri} según (5.19) y (5.20), respectivamente, y modificamos $N_{\mathrm{R}i\mathrm{D}_{\mathrm{salto}\ r}}$ sumándole $Q_{\mathrm{dist}}/L_{\mathrm{R}ri\mathrm{D}_{\mathrm{salto}}}$.

El procedimiento de unión de las áreas asociadas a las referencias requerirá una sola iteración sobre el conjunto de referencias si éstas las ordenamos bien en orden creciente bien en orden decreciente de su dirección de inicio Q_{R_r}. Además el ordenamiento permite simplificar los pasos del algoritmo donde se debe comparar Q_r con Q_{R_r}.

5.3. Anidamientos no perfectos y reuso de datos entre bucles

Las aplicaciones reales no están formadas por un único conjunto de bucles perfectamente anidados, sino que constan de muchos conjuntos de bucles los cuales pueden tener en cada nivel varios bucles a su vez. Por otra parte, las matrices serán accedidas en diversos grupos de bucles, habiendo una probabilidad de acierto en el reuso de líneas previamente accedidas. Aunque no se ha implementado el análisis automático de estos códigos, si podemos indicar la forma de modelarlos.

En nuestro modelo, basado en la extracción de la fórmula del número de fallos para una referencia en un nivel dado de un bucle y en el cálculo de

```
DO I(j+1)=1, N(j+1), S(j+1)
  DO Ij0=1, Nj0, Sj0
    ...
    A(fA01(IA01), ..., fA0dA(IA0dA))
    ...
  END DO
  ...
  DO Ij1=1, Nj1, Sj1
    ...
    A(fA11(IA11), ..., fA1dA(IA1dA))
    ...
  END DO
  ...
  ...
  DO Ijn=1, Njn, Sjn
    ...
    A(fAn1(IAn1), ..., fAndA(IAndA))
    ...
  END DO
END DO
```

Figura 5.8: Código con múltiples referencias a una matriz

la probabilidad de fallo en el acceso a una línea de la matriz estudiada en ese bucle, consideramos que dentro de un conjunto de bucles perfectamente anidados sólo puede haber referencias a una matriz dada en uno de los niveles. En general, cuando esto no es así en un código, sólo hay acceso a la matriz en el mayor de los niveles en los que se la referencia, efectuándose las referencias en los niveles inferiores mediante el acceso a un registro. De esta forma, desde el punto de vista de los accesos a la caché, nuestra hipótesis es cierta.

Así pues, la existencia de anidamientos no perfectos en el nivel donde se encuentra la referencia que estamos analizando o los inferiores, influirá únicamente en el cálculo del vector de área de las interferencias. Se analizarán las referencias internas a dichos bucles para calcular la forma de las regiones a que acceden en las matrices correspondientes y se las intentará fusionar con regiones de las mismas matrices que hubieran podido ser accedidas en otros bucles internos al nivel estudiado, de forma similar a como se ha comentado en el apartado anterior.

Al considerar niveles superiores, cabe la posibilidad de que se acceda a la matriz que se está estudiando en varios de los bucles que contienen. Esta situación puede describirse en general mediante el código de la figura 5.8.

En este caso, y dado que el análisis se efectúa partiendo del bucle más interno que contiene la referencia hacia afuera, aplicaríamos las estrategias descritas hasta el momento para calcular el número de fallos sobre las referencias a la matriz `A` en los anidamientos de los bucles jk con $k = 0, 1, \ldots, n$. Hemos visto en la sección 5.1.2 que el cálculo de la expresión del número de fallos para una referencia R en un nivel de anidamiento dado i, $F_i(R, p)$ se construye a partir de los parámetros de dicho bucle (N_i y L_i), de la expresión para el número de fallos en el nivel inferior, $F_{i-1}(R, p)$, y de $S_0(\mathtt{A}, i, 1)$, el vector de área de interferencia sobre la estructura de datos en una iteración del bucle i. En este caso, al contener el nivel $j + 1$ no un único anidamiento de nivel j sino n, el cálculo de F_{j+1} requerirá unas consideraciones diferentes según el bucle al que pertenezca cada referencia.

Para las referencias R en el interior de los anidamientos jk con $k > 0$, se adopta como valor para el número de fallos que generan durante una iteración completa en el bucle de nivel j la siguiente constante:

$$F_j(R, p) = F_{jk}(R, (S(\mathtt{A}, j(k-1), N_{j(k-1)}/L_{j(k-1)}) \cup \dot{S}(j(k-1), jk))_0) \tag{5.23}$$

es decir, el número de fallos será el que viene dado por la expresión para el bucle correspondiente estimando la probabilidad de fallo al acceder a una línea de la estructura de datos en el momento de entrar en dicho bucle. La estimación más sencilla es aproximar dicha probabilidad por la interferencia generada durante la ejecución del bucle precedente $S(\mathtt{A}, j(k-1), N_{j(k-1)}/L_{j(k-1)})$ más la de los posibles bucles que pudiera haber entre el $j(k-1)$ y el jk en los que no se referenciase la estructura de datos estudiada. Hemos introducido la notación $\dot{S}(j(k-1), jk)$ para representar el vector de área de interferencia cruzada generado por los accesos que pueda haber entre la finalización de la ejecución del bucle $j(k-1)$ y el de inicio del bucle jk.

Esta aproximación es buena si el anidamiento jk contiene al menos un bucle de reuso para las referencias a la estructura `A` o si las estructuras referenciadas y el orden de acceso a sus componentes (en especial los de `A`) es similar en los bucles $j(k-1)$ y jk. En caso contrario la precisión podría verse seriamente afectada, requiriéndose un análisis de la probabilidad como los empleados por ejemplo en el modelado de la trasposición de una matriz dispersa, donde se calculaba la probabilidad de acierto en el reuso para cada línea del vector `RT` desde su último acceso en el bucle precedente hasta el

primer punto de acceso en el bucle analizado. Esta última técnica sólo hemos podido aplicarla por ahora de forma manual.

Para las referencias en el interior del anidamiento $j0$, $F_j(R,p)$ no puede venir dado por una constante, pues al ser el primero contenido en el nivel $j+1$ dependerá de las probabilidades en bucles externos o precedentes. En este caso será simplemente $F_{j0}(R,p)$. No obstante, al calcular la probabilidad de acierto en los reusos en el interior del bucle $j+1$, se estimará $S(\mathtt{A}, j+1, 1)$ como $S(\mathtt{A}, jn, 1) \cup \dot{S}(jn, j0)$, ya que en realidad sólo será necesario tener en cuenta las interferencias desde el último de los anidamientos donde la matriz o vector `A` resulta accedido. Las observaciones sobre la validez de la aproximación son las mismas que en el caso anterior.

5.4. Verificación del modelado automático

Se ha implementado un analizador autómatico basado en este modelo que recibe la descripción del código mediante llamadas a funciones. La implementación acepta vectores y matrices bidimensionales accedidas en bucles perfectamente anidados y proporciona funciones para permitir el modelado de códigos con anidamientos no perfectos permitiendo combinar las funciones de número de fallos y vectores de áreas que corresponden a los anidamientos perfectos que puedan contener, que son los que se pueden calcular sin intervención del usuario.

Los parámetros de entrada al programa son los siguientes:

- Descriptores de la caché: C_s, L_s y K.
- Profundidad de anidamiento, y para cada nivel:
  - número de iteraciones.
  - paso.
  - bucle del que depende en un *blocking*, en caso de haberlo.
- Número de estructuras de datos empleadas, y para cada una:
  - Dirección base.
  - Número de dimensiones.
  - Tamaño de cada dimensión.
- Número de referencias, y para cada referencia:

- Tipo de acceso (lectura o escritura), aun cuando ambos se modelen de la misma forma al asumir una caché de postescritura. Este dato sería de interés al modelar cachés de escritura directa, posibilidad que el modelo soporta aplicando las modificaciones explicadas en el apartado 3.7.8.
- Identificador de la estructura de datos a que está asociado.
- Indicador del bucle al que está asociado la variable que indexa cada dimensión.
- Valor de la constante que puede estar sumada a la variable en la función que indexa cada dimensión.

Como puede observarse este sencillo analizador no permite, por ejemplo, espeficicar el orden en el que se realizan las referencias en un cierto nivel de anidamiento, constantes para multiplicar las variables que controlan las dimensiones de las matrices en las referencias o anidamientos no perfectos, aunque permite un modelado mixto de éstos. No obstante, soporta una buena parte de los códigos densos, y como hemos comentado, estas tres limitaciones no requieren modificaciones del modelo de análisis automático propuesto, sino del analizador concreto que hemos construido. El motivo de las limitaciones de la implementación es que sólo se ha realizado a fin de demostrar la validez del modelo de análisis, no de aplicarla en un entorno real. Nuestro objetivo es integrar el modelado automático en un entorno real de análisis de códigos, el cual es capaz de proporcionar los datos que utiliza la implementación actual y los que aún no soporta sin requerir ningún tipo de intervención del usuario, y de forma que dicho entorno sea capaz de adoptar decisiones sobre la forma del código, las localizaciones de las estructuras de datos empleadas, etc. en función de los datos que proporcione el análisis.

El analizador así construido se ha aplicado a seis códigos densos a fin de verificar su validez:

- el producto matriz densa-matriz densa con orden JIK (figura 5.9).
- un código de Stencil (figura 5.10).
- el método de resolución de ecuaciones diferenciales parciales de Jacobi con acceso por columnas (figura 5.11).
- el producto matriz densa-matriz densa con orden IKJ con un *blocking* sobre las dimensiones J y K (figura 5.12).

```
DO J=0, N-1
  DO I=0, N-1
    R=0.0
    DO K=0, N-1
      R=R+A(I,K)*B(K,J)
    ENDDO
    D(I,J)=R
  ENDDO
ENDDO
```

Figura 5.9: Producto matriz densa-matriz densa con orden JIK.

```
DO I=1, N-1
  DO J=1, N-1
    A(J,I) = A(J,I+1)
             + B(J,I) + B(J+1,I)
             + C(I,J) + C(I+1,J)
  ENDDO
ENDDO
```

Figura 5.10: Código de Stencil.

- una extensión del código anterior añadiendo anidamientos no perfectos al hacer una copia con trasposición del bloque a multiplicar de la matriz densa (figura 5.13).
- un código de resolución de ecuaciones diferenciales parciales simples extraído del programa Ocean que utiliza el método de Gauss-Seidel accediendo por filas (figura 5.14).

A fin de simplificar la validación hemos condiderado $r = 1$ y matrices cuadradas de orden N. Para los tres primeros códigos se comprobaron casi mil combinaciones de los parámetros de entrada con valores de $N = 25$ a 400, y 2400 para los dos códigos con *blocking* usando matrices de tamaño $N = 200$ o $N = 400$. Para cada combinación se realizaron veinte simulaciones modificando en cada una las direcciones base de las matrices, excepto para las matrices grandes del código producto matriz densa-matriz densa optimizado, donde se efectuaron quince simulaciones debido a su elevado coste computacional.

```
DO J=2, N-1
  DO I=2, N-1
    VXN(I,J) = (c0 * VXO(I,J) + dty2 * (VXO(I-1,J) + VXO(I+1,J))
             + dtx2 * (VXO(I,J+1) + VXO(I,J-1))
             - dtx * (PO(I,J) - PO(I,J-1)) - c1) * IVX(I,J)
    VYN(I,J) = (c0 * VYO(I,J) + dty2 * (VYO(I-1,J) + VYO(I+1,J))
             + dtx2 * (VYO(I,J+1) + VYO(I,J-1))
             - dty * (PO(I-1,J) - PO(I,J)) - c2) * IVY(I,J)
  ENDDO
ENDDO
```

Figura 5.11: Método de Jacobi bidimensional.

```
DO J2=1, N, BJ
  DO K2=1, N, BK
    DO I=1, N
      DO K=K2, K2+BK
        RA=A(I,K)
        DO J=J2, J2+BK
          D(I,J) = D(I,J) + B(K,J) * RA
        ENDDO
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

Figura 5.12: Producto matriz densa-matriz con *blocking* perfectamente anidado.

```
DO J2=1, N, BJ
  DO K2=1, N, BK
    DO J=J2, J2+BJ
      DO K=K2, K2+BK
        WB(J-J2+1, K-K2+1) = B(K,J)
      ENDDO
    ENDDO
    DO I=1, N
      DO K=K2, K2+BK
        RA=A(I,K)
        DO J=J2, J2+BK
          D(I,J) = D(I,J) + WB(J-J2+1,K-K2+1) * RA
        ENDDO
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

Figura 5.13: Producto matriz densa-matriz con *blocking* y copia con trasposición del bloque.

```
E=0
WHILE (E.EQ.0)
  D=0
  DO I=2, N-1
    DO J=2, N-1
      T = A(I,J)
      A(I,J) = 0.2 * (T + A(I,J-1) + A(I-1,J)
                      + A(I,J+1) + A(I+1,J))
      D = D + ABS(A(I,J)-T)
    ENDDO
  ENDDO
  IF (D/(N*N)<TOL) E=1
ENDWHILE
```

Figura 5.14: Código de resolución de ecuaciones diferenciales parciales mediante el método de Gauss-Seidel.

Dado que en este caso el algoritmo de modelado emplea también como entrada la posición de inicio de las estructuras de datos, utilizaremos como indicador de error para cada combinación de parámetros de entrada (excluyendo las posiciones iniciales), la media aritmética del valor absoluto del error obtenido para cada simulación. Este error se expresa, como hemos venido haciendo, como el porcentaje que la diferencia entre el número total de fallos predichos y el de medidos supone sobre el número total de fallos medidos.

Las medias del error del total de combinaciones probadas para cada código han resultado ser el 2.44 % para el producto matriz densa-matriz densa, 2.68 % para el código de Stencil, 2.46 % para el método de Jacobi bidimensional, 5.79 % para el producto matriz densa-matriz densa con *blocking* y 5.96 % para el producto matriz densa-matriz densa con copia y trasposición del bloque. La desviación típica respectiva media del número de fallos medidos en las simulaciones de cada combinación de los parámetros de entrada modificando las posiciones de inicio de las estructuras de datos para cada algoritmo ha sido 4.7 %, 2.22 %, 2.73 %, 10.65 % y 11.25 %, respectivamente.

Mención aparte merece la simulación del último código, que al emplear una única matriz, no precisa de varias simulaciones para cada combinación de los parámetros de entrada: no puede haber cambios en el número de fallos al modificar la posición de inicio de la matriz, pues sólo hay auto-interferencias. En este caso además tenemos un código en el que un bucle como los modelados por el algoritmo, de tipo `DO`, está incluido en un bucle de tipo `WHILE` cuyo número de iteraciones desconocemos. Por ello hemos modelado el comportamiento del código en una iteración de este último bucle usando más de 7500 combinaciones de los parámetros de entrada y obteniéndose de ellas un error medio de 2.8 %. Posteriormente se efectuaron 2400 simulaciones usando 2, 3, 5, 10 y 25 iteraciones del bucle, consiguiendo un error medio del 3.7 %.

Las tablas 5.1 a 5.6 muestran los datos obtenidos para la validación de los algoritmos probados con algunas de las combinaciones de sus parámetros de entrada. En este caso hemos incluido el tamaño de las matrices en unidades y hemos empleado matrices de dimensiones inferiores a las empleadas en la validación de los códigos dispersos. El motivo ha sido el mayor tiempo de simulación que requieren estos códigos y el mayor espacio que ocupan sus estructuras en la memoria, que permite generar interferencias con tamaños menores en las dimensiones. Hemos usado siempre matrices cuadradas, de forma que N nos da el tamaño de la dimensión de cualquier matriz, exceptuando `WB` en el producto matriz densa-matriz densa más optimizado, que será $B_{\mathrm{J}} \times B_{\mathrm{K}}$.

N	C_s	L_s	K	σ	Δ
50	2	8	2	5.62	4.83
100	16	4	1	22.65	3.30
175	32	4	1	1.48	6.13
200	16	4	1	0.64	0.45
200	8	8	1	1.32	1.59
250	4	8	2	0.95	0.87
250	32	16	2	0.00	0.03
300	1	4	1	0.08	1.26
300	16	4	4	0.00	0.00
375	32	4	4	0.00	0.00
400	8	8	2	0.85	3.01
400	32	8	2	0.75	0.16

Cuadro 5.1: Datos de validación del modelo automatizado para el producto matriz densa-matriz densa con orden JIK.

N	C_s	L_s	K	σ	Δ
50	2	8	2	1.06	4.55
100	16	4	1	1.09	1.31
175	32	4	1	6.51	1.38
200	16	4	1	6.02	0.57
200	8	8	1	0.73	11.34
250	4	8	2	10.87	4.23
250	32	16	2	1.81	2.47
300	1	4	1	2.68	2.48
300	16	4	4	0.00	0.44
375	32	4	4	0.00	0.62
400	8	8	2	1.34	2.13
400	32	8	2	0.67	0.16

Cuadro 5.2: Datos de validación del modelo automatizado para el código Stencil.

N	C_s	L_s	K	σ	Δ
50	2	8	2	5.4	5.62
100	16	4	1	20.09	2.62
175	32	4	1	15.87	1.47
200	16	4	1	38.07	1.41
200	8	8	1	13.49	1.25
250	4	8	2	14.47	0.98
250	32	16	2	3.07	1.10
300	1	4	1	24.63	0.97
300	16	4	4	0.00	0.90
375	32	4	4	0.00	0.72
400	8	8	2	15.60	1.18
400	32	8	2	6.64	0.67

Cuadro 5.3: Datos de validación del modelo automatizado para el método de Jacobi en dos dimensiones.

En la tabla 5.2 pueden apreciarse desviaciones de cierta importancia en alguna de las combinaciones para el código de Stencil. El origen está en la simplificación del cálculo de la probabilidad de auto-interferencia en el acceso a la matriz `C`. La expresión S'_{ra}, desarrollada en el apartado 2.3.5, permite calcular el vector de área de auto-interferencia para regiones separadas por una distancia constante cuando hay un desfase de una palabra en la posición de inicio de las regiones que preceden a la que se está considerando con respecto a la de las regiones que la siguen. Pero esta expresión está desarrollada considerando que el acceso se produce en el último punto de cada región, pues es el que se estudia para calcular el vector de área. Las referencias a la matriz `C` (`C(I,J)` y `C(I+1,J)`) acceden a las posiciones con el mismo índice de fila en las columnas que las siguen y a las posiciones con el índice de fila inmediatamente mayor para las columnas con índice inferior antes de proceder al reuso de la línea que las contiene. Pero el uso de S'_{ra} es sólo realmente adecuado para `C(I+1,J)`, pues es la que referencia la última posición en cada grupo de posiciones consecutivas. El analizador automático aplica la expresión por igual a ambas referencias, consiguiendo así un ahorro en tiempo, ya que en la segunda aplicación no se calcula su valor sino que se extrae de la tabla de valores ya calculados. Como contraprestación se genera un error que puede ser importante en algunos casos.

Los grandes errores que se observan en las tablas 5.4 y 5.5 para algunas combinaciones se deben a la simplicación al estimar el coeficiente de solapamiento de las matrices que tienen bloques como la media de los coefi-

N	B_{J}	B_{K}	C_{s}	L_{s}	K	σ	Δ
200	100	100	16	8	2	0.74	6.76
200	100	100	256	16	2	50.71	4.11
200	200	100	32	8	1	2.35	1.84
200	200	100	128	8	2	3.65	2.73
200	200	100	128	32	2	117.04	67.36
200	50	200	16	4	1	15.70	1.49
200	100	200	32	8	2	15.35	9.29
200	100	200	64	16	1	83.36	3.33
400	100	100	16	8	2	0.09	3.51
400	100	100	256	16	2	5.30	7.45
400	200	100	32	8	1	0.97	1.55
400	200	100	128	8	2	10.77	6.60
400	200	100	128	32	2	21.35	16.94
400	50	200	16	4	1	0.75	0.63
400	100	200	32	8	2	0.45	0.63
400	100	200	64	16	1	26.90	1.66

Cuadro 5.4: Datos de validación del modelo automatizado para el producto matriz densa-matriz densa con orden IKJ y *blocking*.

cientes de solapamiento que se obtienen para cada bloque. La aproximación correcta sería efectuar el cálculo del número de fallos para cada bloque de las matrices con su propio coeficiente, y sumarlos. No hemos adoptado esta política porque el tiempo de ejecución del modelado crecería notablemente y los errores cometidos, aunque importantes, siguen siendo inferiores a la desviación típica del número de fallos medidos.

5.5. Tiempos de simulación versus tiempos de modelado

En el modelado automático, al tiempo requerido para efectuar el modelado en sí es preciso añadirle el del análisis del código y el cómputo de ecuaciones y regiones que en un modelado manual podrían venir insertadas en el código de modelado por el usuario. Por ello, hemos juzgado conveniente comparar aquí el tiempo requerido para efectuar una simulación de uno de estos algoritmos usando nuestro simulador simplificado, con el tiempo que requiere el pequeño analizador que hemos implementado. Los resultados, ex-

N	BJ	BK	C_s	L_s	K	σ	Δ
200	100	100	16	8	2	8.16	6.23
200	100	100	256	16	2	4.80	2.73
200	200	100	32	8	1	8.81	6.88
200	200	100	128	8	2	3.60	2.86
200	200	100	128	32	2	93.42	50.00
200	50	200	16	4	1	5.05	4.62
200	100	200	32	8	2	16.24	12.51
200	100	200	64	16	1	33.54	3.31
400	100	100	16	8	2	6.18	4.48
400	100	100	256	16	2	4.01	4.26
400	200	100	32	8	1	1.50	2.65
400	200	100	128	8	2	15.04	5.82
400	200	100	128	32	2	57.06	44.13
400	50	200	16	4	1	1.52	2.02
400	100	200	32	8	2	7.86	5.55
400	100	200	64	16	1	7.40	7.11

Cuadro 5.5: Datos de validación del modelo automatizado para el producto matriz densa-matriz densa con orden IKJ, *blocking* y copia con trasposición del bloque.

N	Iteraciones	C_s	L_s	K	Δ
50	1	2	8	2	-13.77
100	25	16	4	1	-3.96
175	1	32	4	1	-2.27
200	1	16	4	1	-1.99
200	5	16	4	1	-1.99
250	1	128	8	2	-1.60
300	1	16	4	4	-1.33
300	10	16	4	4	-1.33
400	1	8	8	2	1.24
400	10	8	8	2	1.24
400	1	32	8	2	-1.00
400	25	32	8	2	-1.00
400	1	256	32	4	-1.00
400	25	256	32	4	-1.00

Cuadro 5.6: Datos de validación del modelo automatizado para el código de resolución de ecuaciones diferenciales mediante el método de Gauss-Seidel.

N	C_s	L_s	K	Tiempo simulación	Tiempo modelo
300	32	4	2	16.41	0.004
300	32	4	4	17.17	0.003
300	32	16	2	15.06	0.004
300	128	4	1	14.23	0.022
300	256	8	2	14.19	0.017
500	32	4	2	76.33	0.004
500	1024	16	2	63.77	0.061

Cuadro 5.7: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz densa-matriz densa con orden JIK.

N	C_s	L_s	K	Tiempo simulación	Tiempo modelo
300	32	4	2	0.16	0.004
300	32	4	4	0.18	0.003
300	32	16	2	0.14	0.004
300	128	4	1	0.15	0.022
300	256	8	2	0.16	0.017
500	32	4	2	0.45	0.004
500	1024	16	2	0.41	0.058

Cuadro 5.8: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos del código de Stencil.

traídos del mismo servidor SGI Origin 200 que se utilizó en la sección 3.8, se muestran en las tablas 5.7 a 5.12. Las tablas se han construido de forma análoga a las del apartado precedente. Puede observarse que tamaños grandes de caché aumentan en general el tiempo de computación del análisis. Sin embargo, el aumento del tamaño de las matrices tiene un efecto muy limitado sobre este tiempo.

N	C_s	L_s	K	Tiempo simulación	Tiempo modelo
300	32	4	2	0.47	0.003
300	32	4	4	0.49	0.003
300	32	16	2	0.40	0.003
300	128	4	1	0.43	0.003
300	256	8	2	0.44	0.003
500	32	4	2	1.33	0.003
500	1024	16	2	1.18	0.003

Cuadro 5.9: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos del código de Jacobi bidimensional.

N	B_J	B_K	C_s	L_s	K	Tiempo simulación	Tiempo modelo
300	100	100	32	4	2	13.60	0.006
300	150	150	32	4	2	14.29	0.006
300	100	100	32	4	4	14.21	0.004
300	100	100	32	16	2	14.10	0.005
300	100	100	128	4	1	13.56	0.040
300	100	100	256	8	2	13.36	0.031
500	100	100	32	4	2	62.93	0.006
500	250	250	32	4	2	75.27	0.006
500	250	500	1024	16	2	63.43	0.111

Cuadro 5.10: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz densa-matriz densa con orden IKJ y *blocking*.

N	B_{J}	B_{K}	C_{s}	L_{s}	K	Tiempo simulación	Tiempo modelo
300	100	100	32	4	2	14.72	0.009
300	150	150	32	4	2	15.30	0.009
300	100	100	32	4	4	13.81	0.005
300	100	100	32	16	2	13.32	0.007
300	100	100	128	4	1	14.61	0.058
300	100	100	256	8	2	13.22	0.046
500	100	100	32	4	2	68.00	0.009
500	250	250	32	4	2	80.74	0.009
500	250	500	1024	16	2	67.16	0.165

Cuadro 5.11: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos de producto matriz densa-matriz densa con orden IKJ, *blocking* y copia con trasposición del bloque.

N	Iteraciones	C_{s}	L_{s}	K	Tiempo simulación	Tiempo modelo
300	10	32	4	2	1.40	0.006
300	10	32	4	4	1.44	0.004
300	10	32	16	2	1.33	0.006
300	20	32	16	2	2.66	0.006
300	10	128	4	1	1.31	0.040
300	10	256	8	2	1.31	0.030
500	10	32	4	2	3.97	0.006
500	20	32	4	2	7.83	0.007
500	10	1024	16	2	3.65	0.115

Cuadro 5.12: Tiempos de usuario de la simulación y la ejecución del modelo para algunos ejemplos del código de resolución de ecuaciones diferenciales parciales mediante el método de Gauss-Seidel.

Conclusiones y principales aportaciones

Esta tesis ha cubierto una gran variedad de temas en el campo del modelado analítico del comportamiento de las memorias caché, contemplando el modelado de códigos tanto con patrones de acceso regulares como irregulares y avanzando en la automatización del mismo.

Podemos resumir las principales aportaciones de esta memoria en los siguientes puntos:

- Presentación de una metodología para el modelado analítico de códigos que incluyen patrones de acceso irregulares, aplicando un enfoque probabilístico para estos últimos. Hasta el momento estos patrones sólo habían sido abordados de forma parcial, modelando códigos simples en los que sólo se consideraban algunas de las fuentes de fallos [46]. Nuestro modelado considera todos los tipos de fallos y se ha demostrado su aplicabilidad a códigos más complejos que los manejados previamente en la bibliografía.

- El método se basa en los conceptos de vector de área asociado a una serie de accesos y de la distancia entre los reusos de una línea en término de iteraciones de bucles. Proporcionamos ecuaciones y algoritmos para el cálculo de los vectores de área correspondientes a los patrones de acceso más comunes. A partir de estas ideas aplicamos una estrategia sistemática para el modelado analítico, aplicando las ecuaciones asociadas al patrón de acceso de cada referencia para calcular el vector de área de interferencia que ésta genera entre los reusos de una línea de la estructura que se estudia.

- En el estudio de los patrones irregulares se ha partido del más simple, el generado por datos con una dispersión uniforme, para ir incorporando elementos de complejidad como la distribución en banda, y finalmente,

la distribución en banda con diagonales uniformes pero de diferentes densidades, a la cual pertenecen por ejemplo el 42 % de las matrices de la colección Harwell-Boeing o el 60 % en el caso de las matrices NEP.

- Gracias a su naturaleza totalmente analítica, los modelos obtenidos son totalmente parametrizables, permitiendo obtener mucha información del comportamiento de los algoritmos en función de los distintos parámetros: configuración de la caché, tamaños de las estructuras de datos, etc. y discernir en qué referencias se generan los fallos y las causas de los mismos. Otra ventaja con respecto a las simulaciones es su reducido consumo de recursos computacionales. La diferencia es aún mayor si consideramos los simuladores más habituales, los cuales requieren la generación y almacenamiento en disco de una traza que debe ser luego leida para la simulación de sus efectos sobre una caché.

- El método se ilustra a través de su aplicación a diversos algoritmos considerando gran variedad de patrones de acceso. En concreto, se ha considerado el producto matriz dispersa-vector, matriz dispersa-matriz densa con sus tres posibles ordenamientos, la trasposición de una matriz dispersa y una versión del matriz dispersa-matriz densa altamente optimizada.

- Uno de los puntos débiles del modelado analítico hasta el momento era su baja precisión. El modelado propuesto supera satisfactoriamente este inconveniente, como se ha demostrado en los apartados de validación. Se observan en todos los casos porcentajes de error reducidos, en particular comparándolos con las desviaciones típicas que se pueden producir en el número de fallos de la simulación al modificar las direcciones base de las estructuras de datos. Además los mayores errores suelen producirse en aquellas combinaciones en las que la desviación típica es mayor. Esto sugiere que un aumento del número de simulaciones hará que la estimación del modelo vaya tendiendo asintóticamente a la media de fallos medidos. Esta idea se ha comprobado positivamente en muchos casos.

- Finalmente, se estudia la aplicación de los conceptos de modelado que hemos introducido utilizando códigos irregulares a códigos densos. La mayor regularidad de los patrones de acceso que se dan en este tipo de códigos permite extender el trabajo realizado previamente para presentar una primera aproximación al modelado automático de éstos. Una revisión de la bibliografía descubre que éste es otro campo en el que

se han realizado pocos trabajos hasta la fecha. Nuestra estrategia impone menos restricciones tanto en las configuraciones de las memorias caché como en la estructura de los códigos a modelar que trabajos precedentes [24]. Es posible el análisis de la gran mayoría de códigos densos, permitiendo múltiples referencias por estructura de datos, bucles gobernados por otros bucles (como los utilizados en la técnica de *blocking*) y anidamientos no perfectos bajo ciertas condiciones.

- Un gran subconjunto del enfoque presentado se implementa de forma efectiva en un pequeño analizador que demuestra, al igual que el modelado manual, su efectividad tanto computacional como en términos del margen de error de sus predicciones. Para ello se analizan seis códigos densos que presentan diferentes dificultades.

En cuanto a las líneas de trabajo a seguir en el futuro, resaltamos las siguientes:

- El modelado de la técnica de prebúsqueda, bien en su vertiente hardware bien en la software[36]. En particular, de entre las múltiples estrategias de prebúsqueda hardware, creemos que la más interesante de analizar sería la secuencial en sus variedades más simples [41], [42]; aquellas en las que al acceder a un bloque se genera la prebúsqueda del siguiente. El motivo es que son las únicas con una coste/eficiencia lo suficientemente buena [16] como para ser implementadas (el HP PA7200 [13] implementa la prebúsqueda secuencial con marca).

  Una distinción fundamental que implicará la consideración de la prebúsqueda será la existencia de varios flujos simultáneos de accesos a memoria: los provocados por los fallos en la caché, que exigen que el procesador se detenga hasta que se resuelven, y los correspondientes a las prebúsquedas, que se resuelven en paralelo.

- La extensión del modelado de patrones de acceso irregulares para considerar otras posibles distribuciones de los datos, o en particular, de las entradas de una matriz dispersa.

- La formalización de la expresión de dichas distribuciones así como de las referencias indirectas que generan de forma que se desarrollen técnicas análogas a las explicadas en el capítulo 5 para la automatización del análisis de códigos con patrones irregulares.

- La integración de las técnicas automatizables desarrolladas en entornos de análisis de códigos que las soporten y permitan su aplicación de forma efectiva y rápida a los programas reales.

Bibliografía

[1] A. Agarwal. *Analysis of Cache Performance for Operating Systems and Multiprogramming*. PhD thesis, University of Stanford, Department of Electrical Engineering, May 1987.

[2] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Theory of Computing Systems*, 7(2):184–215, May 1989.

[3] J. M. Anderson, S. P. Amarasinghe, and M. S. Lam. Data and computation transformations for multiprocessors. In *Proc. 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP'95*, pages 166–178, July 1995.

[4] J.-L. Baer and W.-H. Wang. On the inclusion properties for multi-level cache hierarchies. In H. J. Siegel, editor, *Proc. 15th Annual International Symposium on Computer Architecture*, pages 73–80. IEEE Computer Society Press, May 1988.

[5] Z. Bai, D. Day, J. Demmel, and J. Dongarra. A test matrix collection for non-Hermitian eigenvalue problems, release 1.0, Sept. 1996.

[6] P. Bannon and J. Keller. Internal architecture of Alpha 21164 microprocessor. In IEEE, editor, *Digest of papers: Compcon '95: technologies for the information superhighway: March 5–9, 1995, San Francisco, CA, USA*, pages 79–87. IEEE Computer Society Press, March 1995.

[7] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM Press, 1994.

[8] K. Bolland and A. Dollas. Predicting and precluding problems with memory latency. *IEEE Micro*, 14(4):59–67, Aug. 1994.

[9] D. Buck and M. Singhal. An analytic study of caching in computer systems. *J. of Parallel and Distributed Computing*, 32(2):205–214, Feb. 1996.

[10] D. Burger, J. R. Goodman, and A. Kägi. Memory bandwidth limitations of future microprocessors. In *Proc. 23rd Annual International Symposium on Computer Architecure*, pages 78–89. ACM Press, May 1996.

[11] S. Carr. *Memory-Hierarchy Management.* PhD thesis, Dept. of Computer Science, Rice University, Sept. 1992.

[12] D. R. Chakrabarti, N. Shenoy, A. Choudhary, and P. Banerjee. An efficient uniform runtime scheme for mixed regular-irregular applications. In *Proc. 12th ACM Int'l. Conf. on Supercomputing (ICS'98)*, pages 61–68, July 1998.

[13] K. K. Chan et al. Design of the HP PA 7200 CPU. *Hewlett-Packard J.*, pages 25–33, Feb. 1996.

[14] D. W. Clark. Cache performance of the VAX-11/780. *ACM Trans. on Computer Systems*, 1(1):24–37, 1983.

[15] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach.* Morgan Kaufmann, 1998.

[16] F. Dahlgren and P. Stenström. Effectiveness of Hardware-based Stride and Sequential Prefetching in Shared Memory Multiprocessors. In *Proc. First International Symposium on High-Performance Computer Architecture*, Jan 1995.

[17] Digital Equipment Corporation. *pfm - The 21064 Performance Counter Pseudo-Device*, 1995. DEC OSF/1 Manual pages.

[18] R. Doallo, B. B. Fraguela, and E. L. Zapata. Cache probabilistic modeling for basic sparse algebra kernels involving matrices with a non uniform distribution. In *Proc. 24th EUROMICRO Conference (EUROMICRO'98)*, pages 345–348, Aug. 1998.

[19] R. Doallo, B. B. Fraguela, and E. L. Zapata. Direct mapped cache performance modeling for sparse matrix operations. In *Proc. 7th EUROMICRO Workshop on Parallel and Distributed Processing (PDP'99)*, Feb. 1999. to be published.

[20] I. S. Duff, R. G. Grimes, and J. G. Lewis. User's guide for the Harwell-Boeing sparse matrix collection. Technical report, CERFACS, Oct. 1992.

[21] B. B. Fraguela, R. Doallo, and E. L. Zapata. Memory hierarchy performance prediction for blocked sparse algorithms. Submitted.

[22] B. B. Fraguela, R. Doallo, and E. L. Zapata. Cache misses prediction for high performance sparse algorithms. In *Proc. 4th Intl. Euro-Par Conference, LNCS*, volume 1470, pages 224–233, Sept. 1998.

[23] B. B. Fraguela, R. Doallo, and E. L. Zapata. Modeling set associative caches behavior for irregular computations. *ACM Performance Evaluation Review (Proc. SIGMETRICS/PERFORMANCE'98)*, 26(1):192–201, June 1998.

[24] S. Ghosh, M. Martonosi, and S. Malik. Cache miss equations: An analytical representation of cache misses. In *Proc. 11th ACM Int'l. Conf. on Supercomputing (ICS'97)*, pages 317–324. ACM Press, July 1997.

[25] D. Greenley et al. UltraSPARC: the next generation superscalar 64-bit SPARC. In IEEE, editor, *Digest of papers: Compcon '95: technologies for the information superhighway: March 5–9, 1995, San Francisco, CA, USA*, pages 442–451. IEEE Computer Society Press, March 1995.

[26] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, second edition, 1996.

[27] M. D. Hill. *Aspects of Cache Memory and Instruction Buffer Performance.* PhD thesis, Berkeley Computer Science Division, Univ. California, Berkeley, 1987.

[28] D. Hunt. Advanced performance features of the 64-bit PA-8000. In IEEE, editor, *Digest of papers: Compcon '95: technologies for the information superhighway: March 5–9, 1995, San Francisco, CA, USA*, pages 123–128. IEEE Computer Society Press, March 1995.

[29] B. L. Jacob, P. M. Chen, S. R. Silverman, and T.Ñ. Mudge. An analytical model for designing memory hierarchies. *IEEE Transactions on Computers*, 45(10):1180–1194, Oct. 1996.

[30] W. F. King. Analysis of paging algorithms. In *Proc. IFIP Congress*, pages 485–490, Aug. 1971.

[31] A. Lain. *Compiler and Runtime Support for Irregular Computation.* PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[32] A.R. Lebeck and D.A. Wood. Cache profiling and the SPEC benchmarks: A case study. *IEEE Computer*, 27(10):15–26, Oct. 1994.

[33] J. E. MacDonald and K. L. Sigworth. Storage hierarchy optimization procedure. *IBM Journal of Research and Development*, 19(2):133–140, March 1975.

[34] T. Mathisen. Pentium secrets: Undocumented features of the Intel Pentium can give you all the information you need to optimize Pentium code. *Byte Magazine*, 19(7):191–192, July 1994.

[35] MIPS Technologies Inc. *R10000 Microprocessor Technical Brief*, Oct. 1994.

[36] T. C. Mowry. *Tolerating Latency Through Software Controlled Data Prefetching*. PhD thesis, Dept. of Computer Science, Stanford University, March 1994.

[37] T. C. Mowry, M. S. Lam, and A. Gupta. Design and evaluation of a compiler algorithm for prefetching. In *Proc. Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 62–73, Oct. 1992.

[38] J.J. Navarro, E. García, J.L. Larriba-Pey, and T. Juan. Block algorithms for sparse matrix computations on high performance workstations. In *Proc. 10th ACM Int'l. Conf. on Supercomputing (ICS'96)*, pages 301–309, May 1996.

[39] S. Pissanetzky. *Sparse Matrix Technology*. Academic Press, 1984.

[40] R. W. Quong. Expected i-cache miss rates via the gap model. In *Proc. 21st Int'l. Symp. on Computer Architecture (ISCA'94)*, pages 372–383, April 1994.

[41] A. J. Smith. Sequential program prefetching in memory hierarchies. *IEEE Computer*, 11(11):7–21, Dec. 1978.

[42] A. J. Smith. Cache memories. *ACM Computing Surveys*, 14(3):473–53, Sept. 1982.

[43] V. E. Taylor. Sparse matrix computations: Implications for cache designs. In *Proc. IEEE Int'l Conf. on Supercomputing (ICS'92)*, pages 598–607, Nov. 1992.

[44] O. Temam, C. Fricker, and W. Jalby. Evaluating the impact of cache interferences on numerical codes. In *Proc. 1993 International Conference on Parallel Processing*, volume I - Architecture, pages I–180–I–183. CRC Press, Aug. 1993.

[45] O. Temam, C. Fricker, and W. Jalby. Cache interference phenomena. In *Proc. Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 261–271. ACM Press, May 1994.

[46] O. Temam and W. Jalby. Characterizing the behaviour of sparse algorithms on caches. In *Proc. IEEE Int'l. Conf. on Supercomputing (ICS'92)*, pages 578–587, Nov. 1992.

[47] R.A Uhlig and T.N. Mudge. Trace-driven memory simulation: A survey. *ACM Computing Surveys*, 29(2):128–170, June 1997.

[48] E. H. Welbon, C. C. Chan-Nui, D. J. Shippy, and D. A. Hicks. The POWER2 performance monitor. *IBM Journal of Research and Development*, 38(5):545–554, Sept. 1994.

[49] H. A. G. Wijshoff. Implementing sparse BLAS primitives on concurrent/vector processors. In Alan Gibbons and Paul Spirakis, editors, *Lectures on Parallel Computation*, volume 4 of *Cambridge International Series on Parallel Computation*. Cambridge University Press, 1993.

[50] R. P. Wilson, R. S. French, C. S. Wilson, S. P. Amarasinghe, J. M. Anderson, S. W. K. Tjiang, Shih-Wei Liao, Chau-Wen, Tseng, M. W. Hall, M. S. Lam, and J. L. Hennessy. SUIF: an infrastructure for research on parallelizing and optimizing compilers. *ACM SIGPLAN Notices*, 29(12):31–37, Dec. 1994.

[51] K. C. Yeager. The MIPS R10000 superscalar microprocessor — emphasizing concurrency and latency-hiding techniques to efficiently run large, real-world applications. *IEEE Micro*, 16(2):28–40, April 1996.

[52] M. Zagha, B. Larson, S. Turner, and M. Itzkowitz. Performance analysis using the MIPS R10000 performance counters. In ACM, editor, *Proc. Supercomputing '96 Conference*, pages 17–22. ACM Press and IEEE Computer Society Press, Nov. 1996.

[53] Z. Zhang and J. Torrellas. Speeding up irregular applications in shared-memory multiprocessors: Memory binding and group prefetching. In *Proc. 22nd Annual International Symposium on Computer Architecture*, pages 188–199. ACM SIGARCH and IEEE Computer Society TCCA, June 1995.