Cache Behavior Modelling for Codes Involving Banded Matrices *

Diego Andrade, Basilio B. Fraguela, and Ramón Doallo

Universidade da Coruña
Dept. de Electrónica e Sistemas
Facultade de Informática
Campus de Elviña, 15071 A Coruña, Spain
{dcanosa,basilio,doallo}@udc.es

Abstract. Understanding and improving the memory hierarchy behavior is one of the most important challenges in current architectures. Analytical models are a good approach for this, but they have been traditionally limited by either their restricted scope of application or their lack of accuracy. Most models can only predict the cache behavior of codes that generate regular access patterns. The Probabilistic Miss Equation(PME) model is able nevertheless to model accurately the cache behavior for codes with irregular access patterns due to data-dependent conditionals or indirections. Its main limitation is that it only considers irregular access patterns that exhibit an uniform distribution of the accesses. In this work, we extend the PME model to enable to analyze more realistic and complex irregular accesses. Namely, we consider indirections due to the compressed storage of most real banded matrices.

1 Introduction

Memory hierarchies are essential in current architectures, since they cushion the gap between memory and processor speed. Understanding and improving the usage of caches is therefore absolutely necessary for obtaining good performance both in sequential and parallel computers. There are several methods to study the cache behavior. For example, trace-driven simulations [1] provide accurate estimations of the cache behavior but the required simulations have a high computational cost. Hardware counters [2] yield also accurate estimations but the execution of the real code is needed and their use is limited to the architectures where such registers exist. Both techniques provide a summarized characterization of the cache behavior and little insight about the observed behavior is obtained. As a result, it is difficult to benefit from the information generated in order to improve the cache performance.

* This work has been supported in part by the Ministry of Science and Technology of Spain under contract TIN 2004-07797-C02-02 and Xunta de Galicia under contract PGIDIT 03TIC10502PR

Analytical models of the source code [3, 4] are the best suited approach to enable compilers to extract the behavior of the memory hierarchy and guide optimizations based in this understanding. Models can obtain an accurate prediction of the cache behavior based in the analysis of the source code to execute. Their main drawback is their limited scope of application. Most of them are restricted to codes with regular access patterns. There have been few attempts to model irregular codes but they are either non-automatable [5] or quite imprecise [6]. The Probabilistic Miss Equation (PME) model is nevertheless able to analyze automatically codes with irregular access patterns originated by data-dependent conditionals [7] or indirections [8] with a reasonable accuracy. In the case of irregular codes due to indirections, some knowledge about the distribution of the values contained in the structure that produces the indirection is required in order to achieve certain precision in the predictions. Until now the PME model could only model with a reasonable accuracy indirect accesses that follow an uniform distribution, that is, access patterns in which every position of the dimension affected by the indirection has the same probability of being accessed. This model extension was fully automated and integrated in a compiler in [9]. In the present work the PME model is extended to be able to model automatically and precisely an important class of non-uniform irregular access patterns. Namely, we consider the indirections generated by the compressed storage of realistic banded matrices, a very common distribution in sparse matrices [10]. Most banded matrices are composed of a series of diagonals with different densities of nonzeros. This way, we have developed a more general model that considers this kind of distribution. The accuracy of this new extension will be evaluated using well-known matrix collections.

The rest of the paper is organized as follows. Section 2 introduces the basics of the PME model. Then, Section 3 discusses the extended scope of the model and the problems that the modeling of non-uniformly distributed irregular accesses implies. The extension of the model to cover indirections using realistic banded matrices is described in Section 4. Section 5 is devoted to the experimental results. Section 6 briefly reviews the related work. Finally, in Section 7 the conclusions of our work are established.

2 Introduction to the Probabilistic Miss Equations (PME) Model

Our model estimates the number of misses generated by a code studying the behavior of each static reference R separately. Its strategy lies in detecting the accesses of R that cannot exploit reuse in the cache, and the potential reuse distances for those that can. The reuse distance is the interval in the execution between two consecutive accesses to a same line. During the reuse distance other data structures of the program can be accessed that can interfere in the cache with the studied data structure. These reuse distances are measured in terms of iterations of the loops that enclose the reference, and they generate interference miss probabilities that depend on the cache footprint of the regions accessed

during their execution. The estimation of the number of misses generated by the reference is thus a summatory of its first-time accesses to lines (cold misses) and the number of potential reuses it gives place to, multiplied by the interference miss probability associated to their reuse distance (capacity and conflict misses). This summatory is what we call a Probabilistic Miss Equation (PME), and it is built analyzing the behavior of the reference in each nesting level i that encloses it, beginning in the innermost one and proceeding outwards. In each level the model builds a partial PME F_{Ri} that captures the information on the reuses with reuse distances associated to this loop. Namely, the model calculates the number of different sets of lines (SOLs) that the reference may access during the execution of the loop, the potential reuses for those SOLs, with their corresponding reuse distance, and also the probability those reuses actually take place. A SOL is the set of lines that R can access during one iteration of the loop. In the innermost loop that contains R, each SOL consists of one line. In outer loops, it consists of the set of lines that R can access during a whole execution of the immediately inner loop. Obviously, the first-time accesses to each SOL during the execution of the loop i cannot exploit the reuse within the loop that F_{Ri} captures, but this does not necessarily turn them into misses. These accesses could enjoy reuses with reuse distances associated to outer loops, or to previous loops, when non-perfectly nested loops are considered. As a result, every PME F_{Ri} needs as input for its evaluation a representation Reg of the memory region accessed since the immediately previous access to any of the SOLs that R references in loop i. Notice that given this reasoning, the number of misses generated by each first access to a SOL found in nesting level i is given by the evaluation of $F_{R(i+1)}$, the PME for the immediately inner level, providing as input the memory region Reg. Similarly, the number of misses generated by the attempts to reuse SOLs in level i will be given by the evaluation of $F_{R(i+1)}$ providing as input the memory region for the reuse distance that F_{Ri} estimates. This way, F_{Ri} is built recursively in terms of $F_{R(i+1)}$.

The calculation of the memory regions accessed during a reuse distance is not covered in this paper due to space limitations (see [8] for more information). Still, it is worth to comment that the PME model maps the regions into a mathematical representation consisting of a vector V of $k+1$ probabilities, where k is the associativity of the cache, called area vector. The first element of this vector, V_0 is the probability a cache set has received k or more lines from the region. Each element V_s of the remaining $k-1$ elements of the area vector contains the probability a given cache set has received exactly $k-s$ lines. In the calculation of this area vector the total cache size C_s, line size L_s and degree of associativity K of the cache are taken into account.

In the innermost loop that contains a reference, the recurrence of PMEs finishes defining $F_{R(i+1)}(Reg)$ as the first component of the area vector associated to Reg. The reason is that in the innermost loop containing R, Reg is the set of regions accessed since the latest access to the line, and if the cache has a LRU replacement policy, the condition for the attempt of reuse to fail is that k or

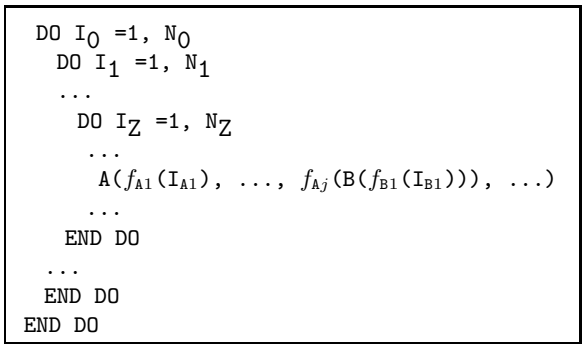


Fig. 1. Nested loops with structures accessed using indirections.

more different lines have been mapped to the set during the reuse distance. This is exactly the probability represented by the first element of the area vector.

On the other hand, the PME for the outermost loop is evaluated using an area vector such that its component 0 is one. The reason is that this vector will only affect those accesses that cannot exploit any reuse within the code, which means that they are the cold misses. Using an area vector V with $V_0 = 1$ for them ensures that the model will predict these accesses result in misses.

3 Scope of Application of the PME Model

Figure 1 shows the kind of codes covered by the PME model extended to cope with irregular access patterns due to indirections [8]. It is a set of perfectly or non-perfectly nested loops in which the number of iterations of the loops must be known at compile-time and the indexing of the data structures must be done using either affine functions of the loop indexes or across indirections. In [8] the values generated by the indirection had to follow an uniform distribution, that is, every position along the indexed dimension had to have the same probability of being accessed. Unfortunately this situation is not very common. In this work we relax this restriction for an important class of codes, namely those that operate with sparse banded matrices, by means of new PMEs.

As for the hardware, the PME model is oriented to caches with LRU replacement policy, allowing arbitrary sizes, block sizes and associativities.

3.1 Complexity of PMEs for Irregular Access Patterns

The equations for references with regular access patterns are relatively simple because all the accesses that can result in a cold miss have an unique interference probability, and a different unique interference probability is applied for the accesses that can result in an interference miss, as all the reuses have the same constant reuse distance.

```
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
```

Fig. 2. Sparse matrix-vector product

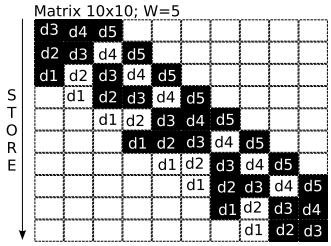


Fig. 3. Banded sparse matrix

In an irregular pattern, every access has a set of different possible reuse distances, each with an associated interference probability. PMEs weight the prediction of misses for each potential reuse distance with the probability that the considered reuse attempt happens. If the distribution of the accesses is uniform, the same set of interference regions can be used for all the accessed lines and they all have the same probability of reuse associated to each reuse distance. When this is not the case, that is, when different lines have different probabilities of being accessed, a different set of interference regions must be calculated for each accessed line, and different lines will have different probabilities of reuse for the same reuse distance.

We will illustrate these ideas with the code in Figure 2, which performs the product between a sparse matrix stored in CRS format [11] and a vector, and which is part of SPARSKIT [12]. The CRS (compressed row storage) format stores sparse matrices by rows in a compressed way using three vectors. One vector stores the nonzeros of the sparse matrix ordered by rows, another vector stores the column indexes of the corresponding nonzeros, and finally another vector stores the position in the other two vectors in which the data of the nonzeros of each row begins. In our codes we always call these vector `A`, `C` and `R` respectively. The innermost loop of the code in Figure 2 performs the product between vector `X` and row `I` of the sparse matrix. In this code reference `X(C(J))` performs an irregular access on vector `X` only in the positions in which the matrix row contains nonzeros. Let us suppose that the sparse matrix that is being multiplied is a banded matrix like the one shown in Figure 3, in which the $W = 5$ diagonals that constitute its band have been labeled. During the processing of each row of the sparse matrix, a maximum of W different elements of `X` will be accessed. Each one of these W elements has a different probability of being accessed that depends on the density of the corresponding diagonal in the banded matrix. The set of elements eligible for access is displaced one position in the processing of each new row. Also, each element of `X` will be accessed a maximum of W times during the execution of the code, as a maximum of W rows may have nonzeros in the corresponding column. Interestingly, the probability of access is not uniform along those W rows. For example, every first potential access during the processing of this matrix in this code will take place for sure, while every second potential access to an element of `X` will happen with a probability of 30%.

This is because all the positions in the fifth diagonal (d_5) keep nonzeros, while in the fourth diagonal (d_4) of the band 3 out of its 9 positions keep nonzeros, which is a density of nonzeros of 30%

The situation depicted in our example is clearly more common than the one modeled in our previous work [8], in which we only considered irregular access patterns which had an uniform probability of access for each element of the dereferenced data structure, and in which such probability did not change during the execution of the code. It is very usual that the diagonals of banded matrices have different densities, with the distribution of the nonzeros within each diagonal being relatively uniform. As a result, we have extended our model to cope with this important class of matrices, which enables to model automatically and accurately the cache behavior of codes with irregular access patterns in the presence of a large number of real sparse matrices, as the evaluation proves. We will characterize the distribution of nonzeros in these matrices by a vector $\vec{d}$ of W probabilities where d_i contains the density of the $i-th$ diagonal, that is, the probability a position belonging to the $i-th$ diagonal of the band contains a nonzero. This extension can be automated using a compiler framework that satisfies its information requirements, such as the one used in [9]. The vector $\vec{d}$ of diagonal densities is the only additional information we need in this work with respect to [9]. These values are obtained from an analysis of the input data that can be provided by the user, or obtained by means of runtime profiling.

4 PME Model Extension for Non-Uniform Banded Matrices

As explained in Section 2, the PME model derives an equation F_{Ri} that calculates the number of misses for each reference R and nesting level i. This PME is a function of input memory regions calculated in outer or preceding loops that are associated to the reuses of the sets of lines (SOLs) accessed by R in loop i whose immediately preceding access took place before the loop began its execution. The uniformity of the accesses in all our previous works allowed to use a single region Reg for this purpose, that is, all the SOLs had the same reuse distance whenever a loop began. This happened because all the considered lines had uniform probabilities of access, and thus they also enjoyed equal average reuse distances and miss probabilities. The lack of uniformity of the accesses makes it necessary to consider a separate region of interference for each SOL. Thus we extend the PMEs to receive as input a vector $\vec{Reg}$ of memory regions. The element Reg_l of this vector is the memory region accessed during the reuse distance for what in this level of the nest happen to be first access to the l-th SOL that R can access. Another way to express it is that Reg_l is the set of memory regions that could generate interferences with an attempt to reuse the l-th SOL right when the loop begins its execution. This way, $\vec{Reg}$ has as many elements as SOLs defines R during the execution of the considered loop.

The shape of PME F_{Ri} depends on the access pattern followed by R in loop i. The PME formulas for references following a regular access pattern and ref-

erences following irregular access patterns due to indirections with a uniform distribution have been presented in [8]. A simple extension was also proposed in [8] to support accesses generated by the processing of banded matrices with an uniform distribution of the entries inside the band by applying small modifications to the formulas of indirections with uniform distributions. This section contains a description of the formulas we have developed for references with irregular access patterns generated by indirections due to the compressed storage of banded matrices in which the distribution of non-zeros within the band is not uniform. In the remaining, the array accessed using an indirection will be known as the base array while the array that is used to generate the values of the index of the indirection will be known as the index array. A different formula will be applied depending on whether the values read from the index array are known to be monotonic or not. They are monotonic when, given two iterations of the current loop i and j and being $f(i)$ and $f(j)$ the values generated by the index array in these iterations, for all $i \leq j$ then $f(i) \leq f(j)$ or for all $i \leq j$ then $f(i) \geq f(j)$. When the index values are known to be monotonic a more accurate estimation can be obtained because we known that if our reference R reuses a SOL of the base array in a given iteration, this SOL is necessarily the one accessed in the previous iteration of the loop.

4.1 PME for irregular monotonic access with non-uniform band distribution

If we assume that the nonzeros within each row have been stored ordered by their column index in our sparse matrix in CRS format, reference `X(C(J))` generates a monotonic irregular access on the base array `X` during the execution of the innermost loop in Figure 2. Let us remember that the index array `C` stores the column indexes of the nonzeros of the row of the sparse matrix that is being multiplied by `X` in this loop.

The general formula that estimates the number of misses generated by a reference R in nesting level i that exhibits an irregular monotonic access with a non-uniform band distribution is

$$F_{Ri}(\vec{Reg}) = \left(\sum_{l=0}^{L_{Ri}-1} p_{i(lG_{Ri})} F_{R(i+1)}(Reg_l) \right) + \left(\sum_{l=1}^{W} d_l - \sum_{l=0}^{L_{Ri}-1} p_{i(lG_{Ri})} \right) F_{R(i+1)}(IntReg_{Ri}(1)) \quad (1)$$

The interference region from the outer level is different for each set of lines (SOL) accessed and it is represented as a vector $\vec{Reg}$ of L_{Ri} different components, where L_{Ri} is the total number of different SOLs of the base array `A` that R can access in this nesting level. L_{Ri} is calculated as $\lceil W/G_{Ri} \rceil$ being W the band size and G_{Ri} is the average number of positions in the band that give place to accesses of R to a same SOL of the base array `A`. This value is calculated as $\lceil L_s/S_{Ri} \rceil$, being $S_{Ri} = \alpha_{Rj} \cdot d_{Aj}$ where j is the dimension whose index depends on the loop

variable $\mathtt{I}_i$ through the indirection; L_s is the cache line size; α_{Rj} is the scalar that multiplies the index array in the indexing of A, and d_{Aj} is the cumulative size[1] of the j-th dimension of the array A referenced by R.

If we consider reference X(C(J)) in Figure 2, while processing the matrix in Figure 3, with a cache line size $L_s = 2$, in the innermost level $d_{A1} = 1$ and $\alpha_{R1} = 1$. Each $G_{Ri} = 2$ consecutive positions in the band give place to accesses to the same SOL of X. Consequently, since $W = 5$, the number of different SOLs of X accessed would be $L_{Ri} = \lceil 5/2 \rceil = 3$.

The vector of probabilities $\vec{p_i}$ has W positions. Position s of this vector keeps the probability that at least one of the diagonals s to $s + G_{Ri} - 1$ has a nonzero, that is, it is the probability they generate at least one access to the SOL of the base array that would be accessed if there were nonzeros in any of these G_{Ri} diagonals. Each component of this vector is computed as :

$$p_{is} = 1 - \prod_{l=s}^{min(W, s+G_{Ri}-1)} (1 - d_l) \tag{2}$$

Let us remember that $\vec{d}$ is a vector of W probabilities, d_s being the density of the $s-th$ diagonal in the band as it is reflected in Figure 3.

In Formula 1 each SOL l of the base array that R can access in nesting level i has a probability $p_{i(lG_{Ri})}$ of being accessed, where lG_{Ri} is the first band that can generate accesses to the $l-th$ SOL. The miss probability in the first access to each SOL l depends on the interference region from the outer level associated to that SOL Reg_l. The remaining accesses are non-first accesses during the execution of the loop, and because the access is monotonic, their reuse distance is necessarily on iteration of the loop. As a result, the interference region will be $IntReg_{Ri}(1)$, the memory region accessed during one iteration of loop i that can interfere with the reuses of R. The number of potential reuses of SOLs by R in the loop is calculated as $\sum_{l=1}^{W} d_l - \sum_{l=0}^{L_{Ri}-1} p_{i(lG_{Ri})}$, where the first term estimates the number of different accesses generated by R during the processing of a row or a column of a band while the second term is the average number of different SOLs that R accesses during this processing.

4.2 PME for irregular non-monotonic access with non-uniform band distribution

A data structure stored in a compressed format, such as CRS [11], is typically accessed using an offset and length construction [13]. In this situation, very common in sparse matrix computations, the knowledge that the values accessed across the indirection follow a banded distribution can be used to increase the accuracy of the prediction using a specific formula. For example, in the code of Figure 2 the reference X(C(J)) accesses the structure X using an offset and length construction. The values generated by the index array C in the innermost loop

[1] Let A be an N-dimensional array of size $D_{A1} \times D_{A2} \times \ldots D_{AN}$, we define the cumulative size for its j-th dimension as $d_{Aj} = \prod_{i=1}^{j-1} D_{Ai}$

are monotonic but the values read across different iterations of the outermost loop are non-monotonic because a different row is processed in each iteration of this loop. When this situation is detected and we are in the presence of a banded matrix, the behavior of the reference in the outer loop can be estimated as

$$F_{Ri}(RegIn) = N_i F_{R(i+1)}(\vec{Reg}(RegIn)) \tag{3}$$

In this formula the N_i iterations in the current nesting level are considered to repeat the same behavior. Although the $W-1$ first and last iterations have a different behavior than the others as for example their band is not W positions wide, we have checked experimentally that the lost of accuracy incurred when not considering this is not significant. This is expected, as usually the band size W is much smaller than N_i, which is the number of rows or columns of the sparse matrix.

An average interference region for each one of the L_{Ri} SOLs accessed in the inner level must be calculated. This average interference region takes account of all the possible reuses that can take place with respect to a previous iteration of the current loop depending on the different possible combinations of accesses to the studied base array. The interference region associated with each possible reuse distance must be weighted with the probability an attempt of reuse with this reuse distance happens before being added in the computation of the average interference region. The expression that estimates the average interference region associated to the $l-th$ SOL that R can access in this loop is,

$$\begin{aligned} Reg_l(RegIn) = & \prod_{z=lG_{Ri}+1}^{W} (1-p_{iz})(RegIn \cup IntReg_{Ri}(W - lG_{Ri} - 1) + \\ & \sum_{s=lG_{Ri}+1}^{W} p_{is} \left(\prod_{z=lG_{Ri}+1}^{s-1} (1-p_{iz}) \right) IntReg_{Ri}(s - lG_{Ri}) \end{aligned} \tag{4}$$

In the previous section we saw that lG_{Ri} is the first diagonal that could generate an access to the l-th SOL in a given iteration and $p_{i(lG_{Ri})}$ the probability of accessing that SOL during the processing of a row or column of the matrix. As the band is shifted one position to the right every row, in general, the probability that the same SOL of the base array is accessed by R m iterations before the current iteration is $p_{i(lG_{Ri}+m)}$. As a result, $\prod_{z=lG_{Ri}+1}^{W}(1-p_{iz})$ calculates the probability that the $l-th$ SOL has not been accessed in any previous iteration of this loop. In this case the interference region is equal to the union of the input region from the outer level and the region associated to the accesses that take place in the $W - lG_{Ri} - 1$ previous iterations. The union of two regions is performed as the union of their associated area vectors. The addition of a region to the average region weighted by its corresponding probability is performed adding the area vector of the region weighted by the corresponding probability to the vector that represents the average region. Regarding the reuses within loop i, the probability that the last access to a SOL took place exactly m iterations ago is calculated multiplying the probability of being accessed in that iteration

```
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
```

Fig. 4. Sparse Matrix - Dense Matrix (IKJ)

$p_{i(lG_{Ri}+m)}$ by the product of the probabilities of not being accessed in any of the iterations between that iteration and the current iteration $\prod_{z=lG_{Ri}+1}^{lG_{Ri}+m-1}(1-p_{iz})$. The interference region associated to this attempt of reuse will be the region covered by the accesses that take place in those m iterations of the current loop. In this equation $L_{Ri} = L_{Rj}$, $G_{Ri} = G_{Rj}$ and the vector $\vec{p_i} = \vec{p_j}$, being j the innermost nesting level of the offset and length construction.

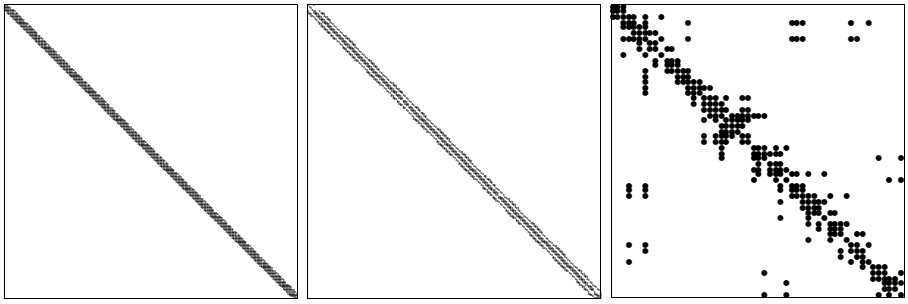
5 Experimental Results

The validation was done applying by hand the PME model to 5 kernels of increasing complexity : an sparse-matrix vector product, see Figure 2, an sparse-matrix dense-matrix product with IKJ (see Figure 4), IJK and JIK order, and a sparse-matrix transposition (omitted due to space limitations). The three sparse-matrix dense-matrix products contain an access to a bidimensional array that contains an indirection in the first dimension, thus they illustrate the correctness of our model when conflicts between columns appear. The sparse-matrix transposition code exhibits particularly complex access patterns, as it has several loop nests with several nesting levels, and it involves references with up to 4 levels of indirection.

The model was validated comparing its predictions with the results of trace-driven simulations. The input data set were the 177 matrices from the Harwell-Boeing [14] and the NEP [15] sets that we found to be banded or mostly banded (a few nonzeros could be outside the band). These matrices represent 52% of the total number of matrices contained in these collections.

The matrices tested are a heterogeneous test set of input data. Some matrices have all their entries uniformly spread along a band, like the AF23560 matrix in Figure 5(a). The LNSP3937 matrix shown in Figure 5(b), has all its values spread along a band of the matrix but not uniformly. Finally, there are some matrices like CURTIS54, shown in Figure 5(c), where not all the values are spread along a band but a significant percentage of them are limited to this area.

Table 1 summarizes data giving an idea of the accuracy of the model. The results were obtained for the benchmarks performing 1770 tests considering 10



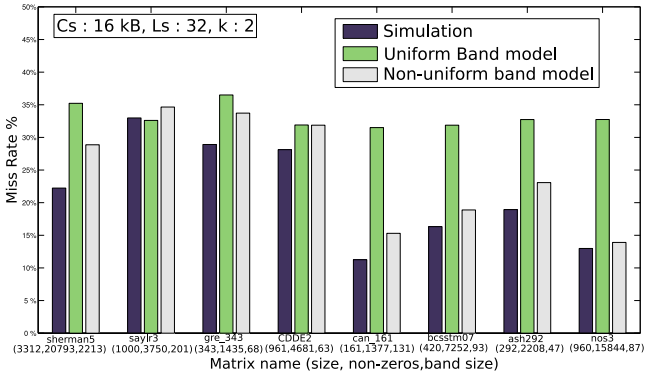
(a) AF23560 M=N=23560 nnz=484256 W=609 (b) lnsp3937 M=N=3937 nnz=25407 W=168 (c) CURTIS54 M=N=54 nnz=291 W=43

Fig. 5. Examples of matrices in the Harwell-Boeing set, M and N stands for the matrix dimension, nnz is the number of nonzeros and W is the band size.

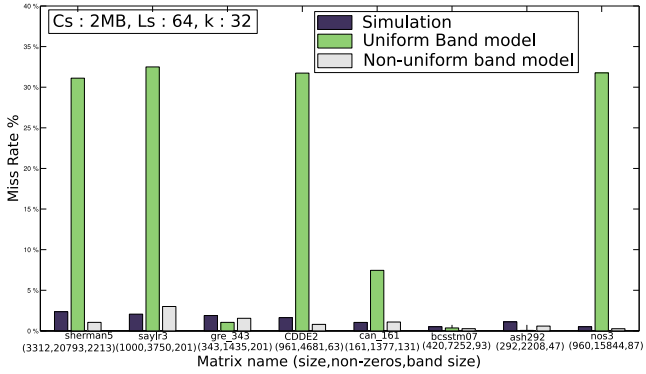
Code	$\overline{MR_{\text{Sim}}}$	$\overline{\sigma_{\text{Sim}}}$	Uniform Bands Model		Non-Uniform Bands Model	
			$\overline{MR_{\text{Mod}}}$	$\overline{\Delta_{MR}}$	$\overline{MR_{\text{Mod}}}$	$\overline{\Delta_{MR}}$
SPMXV	14.00%	0.08%	15.57%	1.80%	14.45%	0.70%
SPMXDMIKJ	27.66%	2.02%	45.62%	26.81%	28.85%	4.19%
SPMXDMIJK	8.62%	0.29%	27.48%	17.23%	10.91%	3.10%
SPMXDMJIK	7.87%	0.43%	10.63%	3.23%	8.36%	0.78%
TRANSPOSE	10.31%	0.33%	11.38%	3.55%	9.52%	3.23%

Table 1. Average measured ($\overline{MR_{\text{Sim}}}$) miss rate, average typical deviation ($\overline{\sigma_{\text{Sim}}}$) of the measured miss rate, average predicted ($\overline{MR_{\text{Mod}}}$) miss rate and the average value $\overline{\Delta_{MR}}$ of the absolute difference between the predicted and the measured miss rate in each experiment.

different cache configurations of each one of the 177 matrices of the Harwell-Boeing and the NEP sets. For each matrix and cache configuration 10 different simulations were performed changing the base address of the data structures involved in each code. In the case of the three orderings of the sparse-matrix dense-matrix product the number of columns of the dense matrix is always a half of its number of rows. The cache configurations have cache sizes (C_{s}) from 16 KBytes to 2 MBytes, line sizes (L_{s}) from 16 to 64 bytes and associativity degrees (K) 1, 2, 4 and 8. Column $\overline{MR_{\text{Sim}}}$ contains the average value of the miss rate simulated in the set of experiments. Column $\overline{\sigma_{\text{Sim}}}$ is the average typical deviation of the miss rate obtained in the 10 simulations performed changing the base address of the data structures. The table compares the precision of the predictions achieved using the simple model for banded matrices assuming an uniform distribution of nonzeros introduced in [8] and the improved model presented in this paper. The table shows for each model, $\overline{MR_{\text{Mod}}}$ the average value of the miss rated predicted, and $\overline{\Delta_{MR}}$ the average value of the absolute value Δ_{MR} of the difference between the predicted and the measured miss rates for each experiment. We use absolute values, so that negative errors are not



(a) Simulation and modeling for a typical level 1 cache configuration



(b) Simulation and modeling for a typical level 2 cache configuration

Fig. 6. Comparison of the miss rates obtained by the simulation, the uniform bands model and the non-uniform bands model during the execution of the sparse matrix-dense matrix product with IJK ordering for several real matrices.

compensated with positive errors. These results show that the improved model is much mode accurate in the presence of real heterogeneous input banded matrices than the original model. The small values of $\overline{\sigma_{\text{Sim}}}$ point out that the base addresses of the data structures play a minor role in the cache behavior.

Figure 6 contains a comparison of the miss rate obtained in the simulation, the miss rate obtained by the uniform bands model and the miss rate obtained by the non-uniform bands model during the execution of the sparse matrix-dense matrix product with IJK ordering using some matrices from the Harwell-Boeing and the NEP collections. The number of columns of the dense matrix used in the multiplication was always one half of the number of rows of the sparse matrix. Figure 6(a) shows the results obtained using a typical level 1 cache configuration, while a typical level 2 cache configuration is used in Figure 6(b). The cache configuration parameters are: C_{s} the total cache size, L_{s} the line size and K the associativity degree. The non-uniform bands model almost always

Architecture	L1 Parameters $(C_{s_1}, L_{s_1}, K_1, \text{Cost}_1)$	L2 Parameters $(C_{s_2}, L_{s_2}, K_2, \text{Cost}_2)$	L3 Parameters $(C_{s_3}, L_{s_3}, K_3, \text{Cost}_3)$
Itanium 2	(16K,64,4,8)	(256K,128,8,24)	(6MB,128,24,120)
PowerPC 7447A	(32K,32,8,9)	(512K,64,8,150)	-

Table 2. Memory hierarchy parameters in the architectures used (sizes in Bytes)

estimates more accurately the miss rate. The difference is bigger in the level 2 cache configuration. The reason for the poor estimations obtained using the uniform bands model is that in matrices with wide bands but in which most of the values are concentrated in a few diagonals, there is a lot of reuse that is not captured by the uniform bands model, as it assumes that the entries are uniformly spread along all the diagonals in the band.

The accuracy of the model and its low computational cost, always less than 1 second in a 2GHz Athlon, makes it very suitable for driving compiler optimizations. As a simple experiment aimed to prove its ability to help optimize codes with irregular access patterns due to indirections, we used its predictions to choose the best loop ordering for the sparse matrix-dense matrix product in terms of the lowest number of CPU detention cycles caused by misses in the memory hierarchy. The prediction is based on a cost function consisting of the addition of the number of misses predicted in each cache level of a real system, weighted by the miss latency for that level. Two very different systems were used for this experiment: an Itanium 2 at 1.5GHz and a PowerPC 7447A at 1.5GHz. Table 2 contains the configurations of the different cache levels in the considered architectures, using the notation (C_s, L_s, K) presented in Figure 6. A new parameter, Cost_i the cost in CPU cycles of a miss in the level i, is also taken into account. Notice that the first level cache of the Itanium 2 does not store floating point data; so it is only used for the study of the behavior of the references to arrays of integers. Also, the PowerPC does not have a third level cache.

Our model always chose the JIK order (see Table 1) in both architectures. The tests were performed using all the banded matrices from the Harwell-Boeing and the NEP collections, in multiplications with dense matrices with 1500 columns. These tests agreed with the predictions of the model: the JIK version was the fastest one in 95.9% and 99.7% of the experiments in both architectures. The codes were compiled using `g77` 3.4.3 with `-O3` optimization level.

6 Related Work

Most of the analytical models found in the bibliography are limited to codes with regular access patterns [4, 3, 16].

The modeling of codes with irregular access patterns has been developed ad-hoc for specific pieces of code. For example, Fraguela et al. [5] proposed a model that obtained a very accurate estimation with a low computation cost but the proposed model was not automatable. In [17], an ad-hoc approach was

proposed but it was limited to direct mapped caches and it did not consider the interaction between different interleaved access patterns.

Some approaches tried to model this kind of codes automatically. Cascaval's indirect accesses model [6] is integrated in a compiler framework, but it is a simple heuristic that estimates the number of cache lines accessed rather than the real number of misses. For example, it does not take into account the distribution of the irregular accesses and it does not account for conflict misses, since it assumes a fully-associative cache. As a result it suffers from limited accuracy in many situations. The modal model of memory [18] requires not only static analysis but also runtime experimentation (potentially thousands of experiments) in order to generate performance formulas. Such formulas can guide code transformation decisions by means of relative performance predictions, but they cannot predict code performance in terms of miss rates or execution time. The validation uses two very simple codes and no information is given on how long it takes to generate the corresponding predictions.

7 Conclusions

We have proposed an automatable extension for the modeling of indirect accesses due to the compressed storage of banded matrices. The model has been validated using codes of increasing complexity and real matrices from the NEP [15] and the Harwell-Boeing [14] collections. Our experiments show that the model reflects correctly the cache behavior of the codes with irregular access patterns due to the operation on banded matrices found in these collections. Such matrices account for 52% of the matrices that these collections contain. This extension achieves higher degrees of accuracy than a previous model of the authors which considered only banded matrices with an uniform distribution of nonzero entries. Besides, the time required to apply the model was less than 1 second in all the performed experiments.

It has been shown that our model can be used as a powerful tool for guiding optimization processes, performing successful experiments in two very different architectures, an EPIC processor Itanium 2 and a superscalar PowerPC 7447A. As future work, we plan to apply the model automatically over a wider range of different codes and to use the model as a guide in more optimization processes.

References

1. Uhlig, R., Mudge, T.N.: Trace-driven memory simulation: A survey. ACM Comput. Surv. **29** (1997) 128–170
2. Ammons, G., Ball, T., Larus, J.R.: Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling. In: SIGPLAN Conf. on Programming Language Design and Implementation. (1997) 85–96
3. Ghosh, S., Martonosi, M., Malik, S.: Cache Miss Equations: A Compiler Framework for Analyzing and Tuning Memory Behavior. ACM Transactions on Programming Languages and Systems **21** (1999) 702–745

4. Chatterjee, S., Parker, E., Hanlon, P., Lebeck, A.: Exact Analysis of the Cache Behavior of Nested Loops. In: In Proceedings of the ACM SIGPLAN'01 Conf. on Programming Language Design and Implementation. (2001) 286–297
5. Doallo, R., Fraguela, B.B., Zapata, E.L.: Cache probabilistic modeling for basic sparse algebra kernels involving matrices with a non uniform distribution. In: In Proceedings of EUROMICRO Conference, Engineering Systems and Software for the Next Decade. (1998) 345–348
6. Cascaval, C.: Compile-time Performance Prediction of Scientific Programs. PhD thesis, Dept. of Computer Science, University of Illinois at Urbana-Champaign (2000)
7. Andrade, D., Fraguela, B.B., Doallo, R.: Analytical modeling of codes with arbitrary data-dependent conditional structures. Journal of Systems Architecture **52** (2006) 394–410
8. Andrade, D., Fraguela, B.B., Doallo, R.: Precise automatable analytical modeling of the cache behavior of codes with indirections (tr-des-00105). Technical report, Dept. of Electronics and System - University of A Coruña (2005)
9. Andrade, D., Arenaz, M., Fraguela, B.B., Touriño, J., Doallo, R.: Automated and accurate cache behavior analysis for codes with irregular access patterns. In: In Proceedings of Workshop on Compilers for Parallel Computers, A Coruña, Spain (2006) 179–193
10. Duff, I., Erisman, A., Reid, J.: Direct Methods for Sparse Matrices. Oxford Science Publications (1986)
11. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J.M., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., der Vorst, H.V.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. Philadalphia: Society for Industrial and Applied Mathematics. (1994)
12. Chow, E., Saad, Y.: Tools and Libraries for Parallel Sparse Matrix Computations. (1995)
13. Lin, Y., Padua, D.: On the automatic parallelization of sparse and irregular fortran programs. In: Languages, Compilers, and Run-Time Systems for Scalable Computers, Pittsburgh (1998) 41–56
14. Duff, I.S., Grimes, R.G., Lewis, J.G.: Users' guide for the Harwell-Boeing sparse matrix collection (Release I). Technical Report CERFACS TR-PA-92-96 (1992)
15. Bai, Z., Day, D., Demmel, J., Dongarra, J.: A test matrix collection for non-Hermitian eigenvalue problems, release 1.0 (1996)
16. Vera, X., Xue, J.: Efficient and accurate analytical modeling of whole-program data cache behavior. IEEE Transactions on Computers **53** (2004) 547–566
17. Ladner, R.E., Fix, J.D., LaMarca, A.: Cache performance analysis of traversals and random accesses. In: In Proceeding of the annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1999) 613–622
18. Mitchell, N., Carter, L., Ferrante, J.: A modal model of memory. In: ICCS '01: Proc. of the Int'l. Conf. on Computational Sciences-Part I. Volume 2073 of Lecture Notes in Computer Science., Springer-Verlag (2001) 81–96