Influence of Cross-Interferences on Blocked Loops: A Case Study with Matrix-Vector Multiply

CHRISTINE FRICKER
INRIA, France
and
OLIVIER TEMAM and WILLIAM JALBY
University of Versailles, France

State-of-the art data locality optimizing algorithms are targeted for local memories rather than for cache memories. Recent work on cache interferences seems to indicate that these phenomena can severely affect blocked algorithms cache performance. Because of cache conflicts, it is not possible to know the precise gain brought by blocking. It is even difficult to determine for which problem sizes blocking is useful. Computing the actual optimal block size is difficult because cache conflicts are highly irregular. In this article, we illustrate the issue of precisely evaluating cross-interferences in blocked loops with blocked matrix-vector multiply. Most significant interference phenomena are captured because unusual parameters such as array base addresses are being considered. The techniques used allow us to compute the precise improvement due to blocking and the threshold value of problem parameters for which the blocked loop should be preferred. It is also possible to derive an expression of the optimal block size as a function of problem parameters. Finally, it is shown that a precise rather than an approximate evaluation of cache conflicts is sometimes necessary to obtain near-optimal performance.

Categories and Subject Descriptors: B.3.0 [**Memory Structures**]: General; C.4 [**Computer Systems Organization**]: Performance of Systems—*modeling techniques*; D.3.4 [**Programming Languages**]: Processors

General Terms: Measurement, Performance

Additional Key Words and Phrases: Blocking, cache conflicts (interferences), cache performance, data locality optimization, numerical codes

1. INTRODUCTION

To date, data locality optimizing algorithms [Eisenbeis et al. 1990; Ferrante et al. 1991; McKinley 1992; Porterfield 1989; Wolf and Lam 1991] have been concerned with decreasing capacity misses using blocking and have mostly ignored the occurrence of conflict misses. However, previous studies [Ferrante et al. 1991; Lam et al. 1991] showed that conflict misses can significantly alter the behavior of blocked algorithms. More precisely, self-interferences in blocked loops [Lam et al. 1991] have been shown to be sensitive to the choice of the optimal block size. A data locality optimization technique which combines tile size optimization and copying has also been proposed [Esseghir 1993] as a way to reduce self-interferences in numerical

This work was funded by the DGXIII ESPRIT BRA III Project APPARC.
Authors' addresses: C. Fricker, INRIA, 78153 Le Chesnay, France; email: Christine.Fricker@inria.fr; O. Temam, PRiSM, University of Versailles, 78000 Versailles, France; email: temam@prism.uvsq.fr; W. Jalby, PRiSM, University of Versailles, 78000 Versailles, France; email: jalby@prism.uvsq.fr.

```
DO j1=0,N-1
  reg = Y(j1)
  DO j2=0,N-1
    reg += A(j2,j1) * X(j2)
  ENDDO
  Y(j1) = reg
ENDDO
```

```
DO jj2=0,N-1,B
  DO j1=0,N-1
    reg = Y(j1)
    DO j2=jj2,min(jj2+B-1,N-1)
      reg += A(j2,j1) * X(j2)
    ENDDO
    Y(j1) = reg
  ENDDO
ENDDO
```

Fig. 1. Blocked and nonblocked matrix vector multiply.

loops. Recently, we have developed a model for evaluating conflict misses in numerical loops [Temam et al. 1994] with the purpose of understanding cache interference phenomena and predicting the cache performance of a numerical loop nest. Three different types of interference misses were distinguished: self-interferences, internal cross-interferences (cross-interferences between two references which subscripts have identical linear expressions), and external cross-interferences (cross-interferences between any two other references). The most frequent and most difficult type of interferences to evaluate are external cross-interferences. We have mentioned in Temam et al. [1993] that two different types of evaluation can be performed: approximate or precise, but up to now we have mostly focused on the approximate evaluation. In this article, precise evaluation of external cross-interferences is shown to be sometimes necessary for computing the near-optimal block size of a numerical loop. Most data locality optimizing algorithms barely deal with the issue of computing the optimal block size. One of the most elaborate treatments of this problem can be found in Eisenbeis et al. [1990], where the computation of the optimal block size sums up to evaluating the number of capacity misses as a function of the block size, and then finding the block size that minimizes this number. The purpose of the article is twofold: provide a detailed illustration of the technique used to derive the precise number of external cross-interference misses and show how the precision of the evaluation of conflict misses can affect the determination of the optimal block size and, further, the performance of the loop.

Position of the Problem. The example used to illustrate the different points developed in this article is the classic numerical algebra primitive: Matrix-Vector multiply and its blocked version (see Figure 1). The target architecture considered is an 8KB *direct-mapped* cache with a line size equal to 32 bytes, which are the parameters of several current processors [Kane and Heinrich 1992; Sites 1992]. All problem parameters are expressed in double-precision floating numbers, i.e., 8 bytes, so that a cache size C_S of 8KB corresponds to $C_S = 1024$, and a line size of 32 bytes to $L_S = 4$.

Notations. m denotes the total number of cache misses. m^t, m^s denote the number of temporal and spatial misses. m^i denotes the number of intrinsic misses. $m(T)$ denotes the total number of cache misses for array T. The notations $m^t(T)$, $m^s(T)$, $m^i(T)$ can also be deduced. Furthermore $m(T_1, T_2)$ denotes the number of misses of T_1 due to interferences with T_2.

Experiments. Throughout the article, the *actual* number of misses is obtained through simulations using a simulator developed for that purpose.

2. ESTIMATING THE NUMBER OF CACHE MISSES

Because of paper length constraints, this section is restricted to studying the external cross-interferences between array A and array X. A treatment of other external cross-interferences in the loop can be found in Fricker et al. [1993]. External cross-interferences basically correspond to the data reused by a reference being flushed from cache by another reference, and the two references have subscripts with distinct linear expressions. The set of data to be reused by the victim reference is called the *reuse set*, and the set of data interfering with this reuse set is called the *interference set*. These sets are defined on the loop level where the reuse occurs. So for arrays X and A in the blocked loop nest, the reuse loop is loop j_1 (for X), and the reuse set (of X) and the interference set (of A) both correspond to a set of B array elements or B/L_S cache lines. The problem sums up to studying the relative cache position of the two sets and to computing the size of their intersection when they overlap. When the intersection size is expressed in cache lines it exactly corresponds to the number of conflict misses between the two references.

2.1 Interferences between X and A

Let us now study the relative cache position of the reuse set of X and the interference set A. The positions of the beginning of these two sets are respectively

$$R_X = x_0 + j_2$$
$$R_A = a_0 + j_2 + Mj_1.$$

Therefore, the relative position of the interference set with respect to the reuse set is the following:

$$R_{XA} = a_0 - x_0 + Mj_1.$$

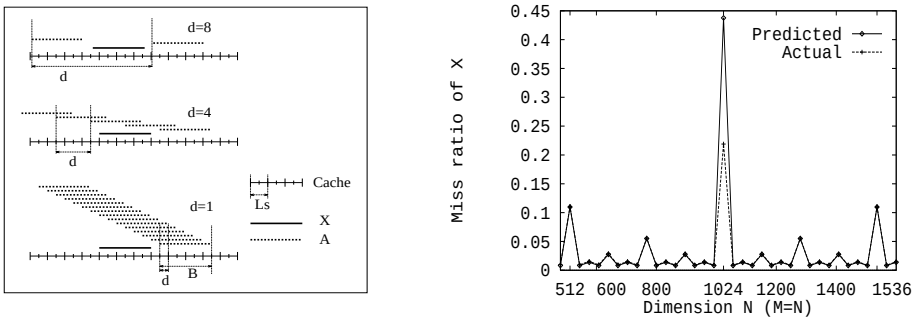
Possible Relative Cache Positions of A and X. The first problem is to find all the possible relative positions of X and A, i.e., all the possible values of R_{XA}.

Since $R_{XA} = a_0 - x_0 + Mj_1$, the possible locations are $(a_0 - x_0 + Mj_1)\ mod\ C_S$. Let $d = gcd\ (M, C_S)$ and $r = (a_0 - x_0)\ mod\ C_S$. Then, $(a_0 - x_0 + Mj_1)\ mod\ C_S = (r + d(M/d)j_1)\ mod\ C_S$. Therefore, the possible positions are all of the form $R_{XA} = (r + \lambda d)\ mod\ C_S,\ \lambda \in \mathbf{Z}$. The set of values of λ corresponding to distinct cache positions is finite. The distance between two consecutive possible cache positions is d, and the number of distinct cache positions is equal to C_S/d.

Cache Positions where Interferences Occur. Let us consider the interval I_λ corresponding to C_S/d consecutive values of λ and defined by $-C_S/2 \leq r + \lambda d \leq C_S/2$. For $\lambda \in I_\lambda$, interferences occur only if $-B \leq r + \lambda d \leq B$,[1] i.e., if the distance in cache between the beginning of the intervals of A and X belongs to $[-B, B]$ (see Figure 2(a)).

The previous inequation can be rewritten as $\lceil (-B - r)/d \rceil \leq \lambda \leq \lfloor (B - r)/d \rfloor$. Let $B = B_d d + b$ with $b = B\ mod\ d$. It is certain interferences occur for $\lambda \in$

[1] It is assumed here that $B \leq C_S/2$.



(a) (b)

Fig. 2. (a) Cross-interferences between A and X. (b) Miss ratio of X.

$[-B_d, B_d - 1]$, while for $\lambda = -(B_d + 1)$ and $\lambda = B_d$, interferences may occur depending on the relative values of b, r, and d (this is due to the *ceiling* and *floor* functions of the above inequation).

Computing the Number of Temporal Interferences. As mentioned in the previous paragraph, the interferences between A and X recur with a period of C_S/d. Therefore, the amount of interferences needs to be computed over one period and then multiplied by the number of periods. An approximate number of periods is $N/(C_S/d)$.

So, in this paragraph, only a chunk of C_S/d iterations is considered, e.g., the interval I_λ. For each value of $\lambda \in I_\lambda$, the distance in cache between the beginning of the intervals of X and A is $|r + \lambda d|$. So, the overlapping (expressed in cache locations) is equal to $(B - |r + \lambda d|)^+$, where $(x)^+ = max\ (x, 0)$. For $\lambda \in [-B_d, -1]$, the overlapping is equal to $(B + r + \lambda d)^+ = B + r + \lambda d$, and for $\lambda \in [0, B_d - 1]$, it is equal to $(B - r - \lambda d)^+ = B - r - \lambda d$. For $\lambda = -(B_d + 1)$, the overlapping is equal to $(B + r - (B_d + 1) \times d)^+ = (b + r - d)^+$, and for $\lambda = B_d$, the overlapping is equal to $(B - r - B_d d)^+ = (b - r)^+$. For any other value of λ such that $-C_S/2 \leq r + \lambda d \leq C_S/2$, the overlapping is equal to 0. Consequently for one *period* of C_S/d iterations the number of cache lines that overlap is equal to

$$\left(\frac{(b + r - d)^+ + (b - r)^+ + \sum_{\lambda=0}^{B_d - 1} B - r - \lambda d + \sum_{\lambda=-B_d}^{-1} B + r + \lambda d}{L_S} \right),$$

and since $\sum_{\lambda=0}^{B_d - 1} B - r - \lambda d + \sum_{\lambda=-B_d}^{-1} B + r + \lambda d = -d{B_d}^2 + 2B_d B = (B^2 - b^2)/d$, the total number of temporal interferences of X due to A is given by

$$m^t(X, A) = \frac{N}{B} \times \frac{N}{\frac{C_S}{d}} \times \left(\frac{(b + r - d)^+ + (b - r)^+ + \frac{B^2 - b^2}{d}}{L_S} \right).$$

An intuitive representation of such interferences is indicated on Figure 2(a) (all intervals of A which do not interfere with X have not been represented).

Average interferences $m^t(X,A)$ can be averaged over all possible values of r which may vary between 0 and $d-1$. The expression of the *average number of interferences* is equal to

$$\frac{N}{B}\frac{N}{\frac{C_S}{d}}\frac{1}{L_S}\frac{1}{d}\sum_{r=0}^{d-1}\left((b+r-d)^+ + (b-r)^+ + \frac{B^2-b^2}{d}\right) = \frac{N^2B}{C_S L_S}.$$

2.2 Total Number of Cache Misses

In this section, the analytical expressions of the different sources of cache misses are presented. In theory, it is not possible, for one array, to add simply all the associated expressions because of possible redundancy between cross-interferences. However, these redundancies have been ignored because they proved to be negligible in most cases.

Array X. Because Y induces a negligible number of spatial interferences on array X, the term $m^s(X,Y)$ does not figure in the expression of $m(X)$. So,

$$m(X) = m^i(X) + m^t(X,A) + m^s(X,A) + m^t(X,Y),$$

with

$$m^i(X) = \frac{N}{L_S}, m^t(X,A) = \frac{N^2B}{C_S L_S}, m^s(X,A) = \frac{N^2 L_S}{C_S}(1-\frac{1}{L_S})^2, m^t(X,Y) = \frac{N^2}{C_S},$$

we obtain

$$m(X) = \frac{N}{L_S} + \frac{N^2B}{C_S L_S} + \frac{N^2 L_S}{C_S}(1-\frac{1}{L_S})^2 + \frac{N^2}{C_S}.$$

The variations of $m(X)$ can be very important, essentially because of the variations of $m(X,A)$. The precision of the above estimate is illustrated in Figure 2(b).

Array Y. The expression of the total number of misses for Y, $m(Y)$, is the following:

$$m^i(Y) + m^t(Y,Y) + \ min\ ((\frac{2C_S-N}{N})^+, 1) \times (m^t(Y,A) + m^t(Y,X)) + m^s(Y,A) + m^s(Y,X)$$

with

$$m^i(Y) = \frac{N}{L_S}, m^t(Y,Y) = \frac{N-(N-2(N-C_S)^+)^+}{L_S}, m^t(Y,A) = \frac{N^2}{BL_S}\ min\ (1, \frac{2B}{d}),$$
$$m^t(Y,X)) = \frac{N^2}{C_S L_S}, m^s(Y,A) = m^s(Y,X) = \frac{N^2}{C}(1-\frac{1}{L_S}),$$

we obtain

$$m(Y) = \frac{N}{L_S} + \frac{N^2}{BL_S}\ min\ (1, \frac{2B}{d}) + \frac{N-(N-2(N-C_S)^+)^+}{L_S}.$$

Array A. Because array A exhibits no temporal locality, the terms $m^t(A,X)$ and $m^t(A,Y)$ do not appear in the expression of $m(A)$. Besides, Y induces a negligible number of spatial misses on array A (the argument is the same as for array X), so the term $m^s(A,Y)$ has been removed as well. So,

$$m(A) = m^i(A) + m^s(A,X),$$

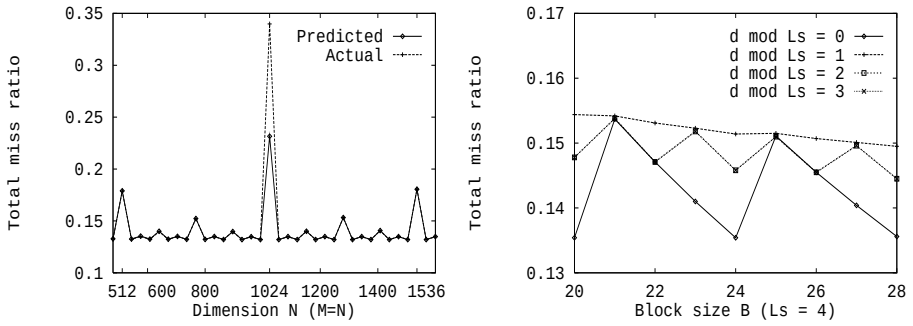
with

$$m^i(A) = \frac{N^2}{L_S}, m^s(A,X) = \frac{N^2 L_S}{C_S}(1-\frac{1}{L_S})^2,$$

we obtain

$$m(A) = \frac{N^2}{L_S} + \frac{N^2 L_S}{C_S}(1-\frac{1}{L_S})^2.$$

Blocked Matrix-Vector Multiply. Regarding the whole primitive, the misses of each array are clearly cumulative; therefore it is safe to assert that the expression



(a) (b)

Fig. 3: (a) Total miss ratio of blocked matrix-vector multiply (r=4). (b) Influence of *semiintrinsic* misses on global performance.

of m, the total number of misses, is the following:

$$m = m(X) + m(Y) + m(A).$$

Because the term $m^t(X, A)$ has a dominant impact on the total miss ratio, the total miss ratio is closely related to the miss ratio of X as the comparison of Figure 3(a) with Figure 2(a) shows.

2.3 Spatial Interferences

Temporal vs. Spatial Interferences. The main source of cache misses are temporal interferences on X due to A: $m^t(X) \simeq (N^2B)/(C_S L_S)$. Similarly, for spatial interferences: $m^s(X) \simeq ((N^2 L_S)/C_S)(1 - 1/L_S)^2$. An upper bound for $m^s(X)$ is $(N^2 L_S)/C_S$. So, if B is large enough $m^s(X) \ll m^t(X)$, i.e., *spatial interferences are negligible with respect to temporal interferences.*

Note that, in opposition to temporal interferences, spatial interferences are independent of B, and therefore *they do not influence the choice of the optimal block size.* As a consequence, spatial interferences will be ignored in the computations of Section 3.

Semiintrinsic Misses. In the nonblocked loop, the reference to A is $R_A = a_0 + j_2 + Mj_1$ with $0 \leq j_1 < N$ and $0 \leq j_2 < N$, i.e., N elements are accessed consecutively; then a stride of M is applied (if $M = N$ all elements are consecutive). In the blocked loop, $R_A = a_0 + j_2 + Mj_1 + Bjj_2$, i.e., the stride of M is applied much more frequently, every B elements. If L_S does not divide B, or if the block of B elements is not aligned on a cache line, some elements of A are loaded that do not belong to this block of B elements, i.e., useless elements. Since such elements will only be used after N iterations of loop j_1 (i.e., they are unlikely to be kept in cache) or have already been used, they breed additional cache misses that can be termed *semiintrinsic* misses.

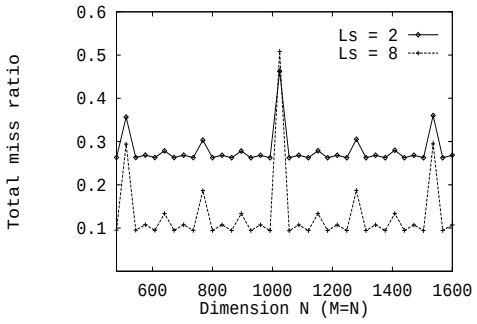


Fig. 4. Influence of L_S on the relative importance of cache interferences.

Even assuming $a_0 \ mod \ L_S = 0$ (the first element of A is aligned on a cache line), semiintrinsic misses occur if $B \ mod \ L_S \neq 0$ (L_S does not divide B) and/or $M \ mod \ L_S \neq 0$ (a block is not always aligned on a cache line). As can be seen in Figure 3(b), the optimal performance of the blocked loop can only be reached if these two conditions are fulfilled.

Also, the influence of L_S on the number of interferences can be seen in Figure 4.

3. OPTIMAL BLOCK SIZE AND OPTIMAL GAIN

The benefit or *gain* of blocking for array T is defined by $G(T) = m_n(T) - m_b(T)$ (where $m_n(T)$ is $m(T)$ for the nonblocked − i.e., standard − loop, and $m_b(T)$ is $m(T)$ for the blocked loop). G is the total gain, i.e., $G = m_n - m_b$. *For all the graphs in this section, the expression of the gain $g = m_n/m_b$ is preferred because it provides the relative instead of the absolute improvement of miss rates due to blocking.* Still $G(T)$ has been used in the computations for the sake of simplicity. Also, in the next sections the optimal block size is denoted B_{opt}.

In Section 3.1, the values of the optimal block size and the gain, as computed by state-of-the-art data locality optimizing algorithms, are provided. In Section 3.2, the average gain (and the associated optimal block size) derived from the expressions of Section 2 is computed. The threshold value of N for which blocking is useful is computed in Section 3.3. The differences between accurate and average evaluation of interferences are highlighted in Section 3.4. In Figure 5 the curves corresponding to the different expressions of the gain are plotted. Each curve is explained in one of the following sections.

3.1 Theoretical Optimal Block Size and Theoretical Gain

To date, the most elaborate method for computing the optimal block size in any loop can be found in Eisenbeis et al. [1990], so we will start from that point. In Eisenbeis et al. [1990], for each reference, the set of data to be reused is called the *reference window*. The principle is to find a block size so that all windows fit in cache, and which minimizes the number of cache misses. In Eisenbeis et al. [1990], only capacity misses are considered.

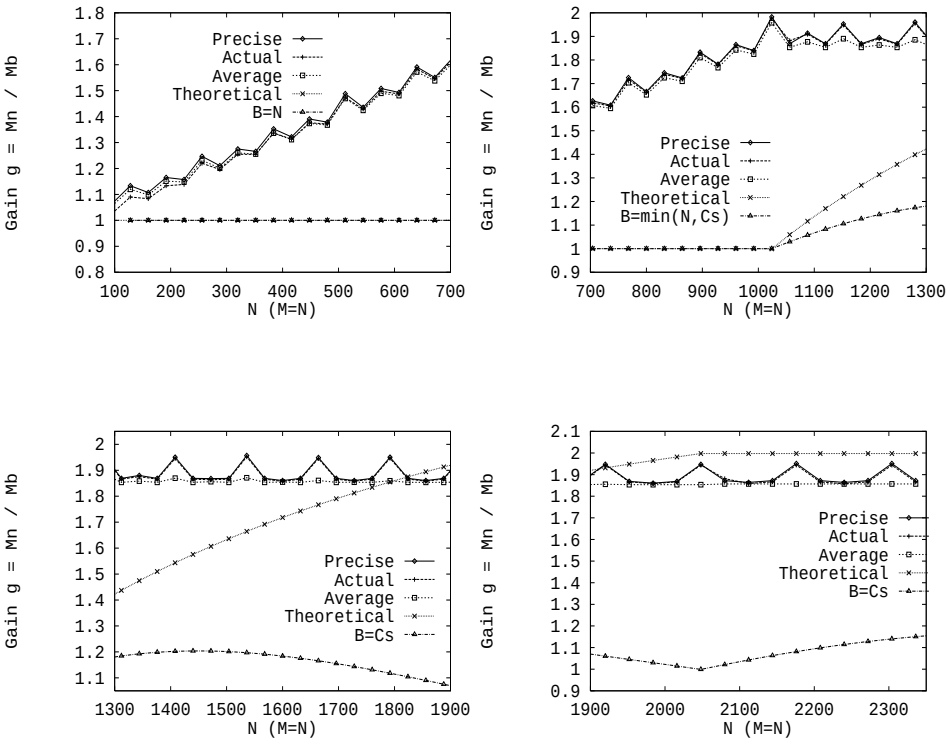


Fig. 5: Optimal gain, predicted precise optimal gain, predicted average optimal gain, theoretical optimal gain.

Let us illustrate this process with blocked matrix-vector multiply. The reference window corresponding to array Y has a size of 1 cache line. For array X it is equal to B/L_S cache lines. And there is no window for array A because it is not reused. No reuse is assumed to occur for arrays to which blocking is not applied, i.e., array Y. So the number of cache misses of array Y is equal to $N/B \times N/L_S$. The number of misses of array A is equal to N^2/L_S (compulsory misses). Finally, since interferences are ignored, and the window of B is assumed to fit in cache, the number of misses of array X is equal to $N/B \times B/L_S$. The optimization problem is then the following:

$$B \leq N$$
$$B \leq C_S$$
$$Minimize \;\; m_b = \frac{N}{B} \times \frac{N}{L_S} + \frac{N^2}{L_S} + \frac{N}{B} \times \frac{B}{L_S} = \frac{N^2}{BL_S} + \frac{N^2}{L_S} + \frac{N}{L_S}.$$

So, in this case, the problem is equivalent to maximizing B under the constraints. If $N < C_S$, then $B_{opt} = N$, and if $N \geq C_S$, $B_{opt} = C_S$, i.e., $B_{opt} = \; min\,(N, C_S)$.

In order to compute the gain, the number of cache misses for the nonblocked

loop nest must be evaluated. Shortly, the number of capacity misses of X in the nonblocked loop nest is equal to $m^t(X,X) = N \times (N - (N - 2(N - C_S)^+)^+)/L_S$. So,

$$\begin{aligned} G = m_n - m_b &= \left(\frac{N-(N-2(N-C_S)^+)^+}{L_S} + \frac{N^2}{L_S} + \frac{N}{L_S}\right) - \left(\frac{N^2}{min\ (N,C_S)L_S} + \frac{N^2}{L_S} + \frac{N}{L_S}\right) \\ &= \frac{N-(N-2(N-C_S)^+)^+}{L_S} - \frac{N^2}{min\ (N,C_S)L_S}. \end{aligned}$$

In the remainder of the article, these values of the optimal block size and the optimal gain are termed the *theoretical optimal block size* and the *theoretical optimal gain*. In Figure 5, it can be seen that the gain obtained with the theoretical optimal block size is very low (lower than 1.2). Besides, the theoretical gain appears to be a strong misprediction of both the actual gain and even the gain obtained with the theoretical block size. The theoretical gain actually corresponds to what "should happen" if blocking was behaving as predicted by the *Window* model, i.e., if capacity misses were removed and there were no interference miss. Incidentally, the theoretical gain indicates the maximum gain that can be theoretically expected, i.e., the *ideal* gain. Let us compute this maximum gain:

$$g = \frac{N \times \frac{N-(N-2(N-C_S)^+)^+}{L_S} + \frac{N^2}{L_S} + \frac{N}{L_S}}{\frac{N^2}{min\ (N,C_S)L_S} + \frac{N^2}{L_S} + \frac{N}{L_S}}.$$

When $N > 2C_S$,

$$g \simeq \frac{2\frac{N^2}{L_S}}{\frac{N^2}{L_S} + \frac{N^2}{C_S L_S}} = \frac{2}{1 + \frac{1}{C_S}},$$

so $g \simeq 2$. The maximum gain that can be expected is 2 (i.e., blocking would divide by 2 the number of cache misses): in the nonblocked loop X exhibits at most N^2/L_S cache misses; and A also exhibits N^2/L_S compulsory misses, while in the blocked loop X ideally exhibits only N/L_S compulsory misses in the best case; and A still exhibits N^2/L_S cache misses.

3.2 Estimated Average Gain and Corresponding Optimal Block Size

3.2.1 *Estimated Average Gain.* For computing the average gain, the expression of the average values of interferences are used. Such average expressions have been derived for both the blocked and the nonblocked loops. Because of paper length constraints, the details of computations have been omitted (see Fricker et al. [1993]).

$N < C_S$.

$$G = m_n - m_b = \left(\frac{N^3}{C_S L_S} + \frac{N^2}{C_S L_S}\right) - \left(\frac{N^2 B}{C_S L_S} + \frac{N^2}{C_S} + \frac{N^2}{BL_S}\ min\ (1, \frac{2B}{d}) + \frac{N^2}{C_S L_S}\right).$$

$C_S \leq N < 2C_S$.

$$\begin{aligned} G &= m_n - m_b \\ &= \left(\frac{N^2}{L_S} + N\right) - \left(\frac{N^2 B}{C_S L_S} + \frac{N^2}{C_S} + \frac{2C_S - N}{N}\frac{N^2}{BL_S}\ min\ (1, \frac{2B}{d}) + \frac{N(2C_S - N)}{C_S L_S} + \frac{2N(N-C_S)}{BL_S}\right). \end{aligned}$$

$2C_S \leq N$.

$$G = m_n - m_b = \left(\frac{N^2}{L_S} + N\right) - \left(\frac{N^2 B}{C_S L_S} + \frac{N^2}{C_S} + \frac{N^2}{BL_S}\right).$$

Because cache interferences are taken into account in the above average estimates and not in the theoretical expressions of Section 3.1, new terms appear, or existing terms are modified. For instance, in the first case ($N < C_S$), the main new term is $N^2B/(C_S L_S)$ which corresponds to temporal interferences between A and X. Because this term is a function of B, it is going to affect the determination of the optimal block size. Indeed, when $N < C_S$, the expression of the theoretical number of misses of the blocked algorithm (see m_b in Section 3.1) only contains one term which depends on B: $N^2/(BL_S)$. Consequently, this term is minimal when the block size is the largest possible; hence $B_{opt} = min\ (N, C_S)$. Now, in the above average expression two terms depend on B: $(N^2B)/(C_S L_S)$ and $(N^2/(BL_S))\ min\ (1, 2B/d)$ which respectively increases and decreases (or is constant) with B. Therefore, the optimal block size is either equal to a tradeoff value or to L_S (see the detailed computations in Section 3.2.2).

The curve *Average* in Figure 5 corresponds to the average optimal gain. It corresponds to the above expressions with $B = B_{opt}$ (except g is used instead of G). It is shown in Section 3.2.2 how to derive the expression of B_{opt} in the different cases.

As can be seen in Figure 5, the precision of the average optimal gain is usually close to the actual optimal gain. Still, when $N > C_S$, the actual gain is periodically slightly higher than the average gain, while the precise estimate of the gain correctly predicts such phenomena (see Figure 5). The main difference between precise and average estimates is that *array base addresses* are considered in the precise estimate. In Figure 5, the base addresses of arrays X and A have been chosen large enough that no intense interference phenomena related to array placement can occur ($r = 52$). But, in Section 3.4, it is shown that array base addresses can sometimes have a major influence on the number of interference misses, in which case the precision of the average estimate can be poor.

3.2.2 *Estimated Optimal Block Size Based on the Average Gain.* Let us first indicate the optimal block size expression obtained in each case and then provide the details of computations.

When $N < C_S$, we obtain $B_{opt} = \sqrt{C_S}$ if $d < \sqrt{C_S}$ and $B_{opt} = L_S$ if $d \geq \sqrt{C_S}$ (recall that $d = gcd\ (M, C_S)$). With the theoretical expression of Section 3.1, we obtain that $B_{opt} = N$.

When $N \geq C_S$, B_{opt} is either equal to $\sqrt{C_S}$ or $\sqrt{2C_S \times (1 - C_S/N)}$ while the theoretical optimal block size is equal to C_S in this case.

Therefore, the theoretical expression of the optimal block size is generally a strong overestimate of the optimal block size, which is confirmed by Figure 5. The theoretical expression of Section 3.1 implies that once an element of X is loaded into the cache, it will not be flushed. Therefore, the only constraint on B is that it must fit in cache. That is why the number of misses of X (N/L_S) does not depend on B. On the other hand, the expressions computed in Sections 2 and 3.2.1 take into account the fact the elements of X can be flushed by elements of A. Consequently, with respect to X, the block size should be selected as small as possible so that *the elements of X can be reused before they can be flushed.* Intuitively, it means the reuse distance should be small enough that the probability an element of X is flushed before it can be reused is negligible. That is why the number of misses of X, $(N^2B)/(C_S L_S)$, increases with B.

In the following paragraphs, it is now shown how the expression of the optimal block size can be derived from the expression of the average gain. G is now considered to be a function of B. It is differentiated along B so that its variations can be analyzed. The optimal value of B, i.e., B_{opt}, is the value that maximizes the gain. The computations are mostly detailed for the first case.

$N < C_S$. Two subcases must be distinguished: $2B/d < 1$ and $2B/d \geq 1$.

$B < d/2$. $\partial G/\partial B = -N^2/(C_S L_S)$, so $\partial G/\partial B < 0$ for this interval of values of B. Therefore the local maximum is reached when B is minimum, i.e., $B_{opt_1} = L_S$. The corresponding value of the gain is $G_{max_1} = G(B_{opt_1})$.

$B \geq d/2$. $\partial G/\partial B = -N^2/(C_S L_S) + N^2/(B^2 L_S)$. Therefore, $\partial G/\partial B > 0$ if $B > \sqrt{C_S}$ and $\partial G/\partial B < 0$ otherwise. Thus, G increases up to the value $B = \ max\ (\sqrt{C_S}, d/2)$ and decreases afterward. So $B_{opt_2} = \ max\ (\sqrt{C_S}, d/2)$. The maximal value of the gain is $G_{max_2} = G(B_{opt_2})$.

The maximal gain for this interval of N is the largest of the two gains, i.e., $G_{max} = \ max\ (G_{max_1}, G_{max_2})$. These values must then be compared to find the global optimum B_{opt} among B_{opt_1} and B_{opt_2}. If $\sqrt{C_S} < d/2$, then $G(L_S)$ and $G(d/2)$ should be compared. We obtain

$$G(L_S) - G(\frac{d}{2}) = \frac{N^2 d}{2C_S L_S} + \frac{N^2}{C_S}.$$

Thus $G(L_S) > G(d/2)$ if $d > 2L_S$, which is assumed. Hence $B_{opt} = L_S$. If $\sqrt{C_S} \geq d/2$, then $G(L_S)$ and $G(\sqrt{C_S})$ should be compared, which gives

$$G(L_S) - G(\sqrt{C_S}) = N^2(\frac{2}{L_S\sqrt{C_S}} - \frac{1}{C_S} - \frac{2}{dL_S}).$$

Thus $G(L_S) > G(\sqrt{C_S})$ if $2/(L_S\sqrt{C_S}) > 1/C_S + 2/(dL_S)$, which is equivalent to $d > \sqrt{C_S}(1 - l/\sqrt{C_S})^{-1} \simeq \sqrt{C_S}$. The optimal block size for this interval of N is $B_{opt} = \sqrt{C_S}$ if $d < \sqrt{C_S}$ and $B_{opt} = L_S$ otherwise.

$C_S \leq N < 2C_S$. The same subcases must be distinguished. We obtain the following:

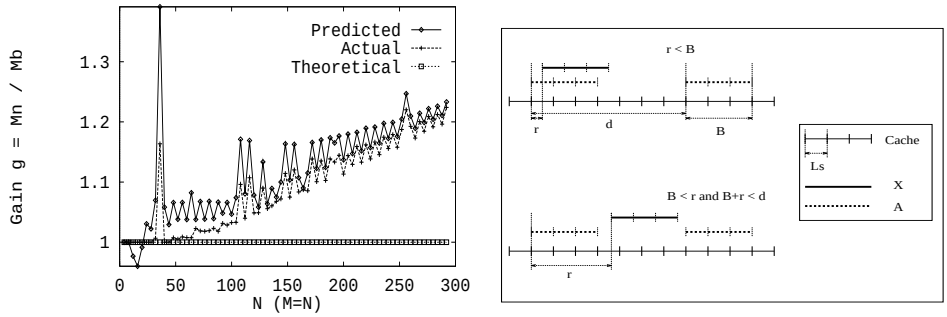
$B < d/2$. $B_{opt_1} = \ min\ (\sqrt{2(N - C_S)C_S/N}, d/2)$.

$B \geq d/2$. $B_{opt_2} = \ max\ (\sqrt{C_S}, d/2)$, and these two local maxima are then compared. Note that $\sqrt{2(N - C_S)C_S/N} < \sqrt{C_S}$; thus three cases must be distinguished, according to the respective positions of d and interval $[\sqrt{2(N - C_S)C_S/N}, \sqrt{C_S}]$. Computations show that the optimal block size is $B_{opt} = \sqrt{C_S}$ if $d < h(N)$ and $B_{opt} = \sqrt{2(N - C_S)C_S/N}$ otherwise, where

$$h(N) = \frac{\frac{2C_S - N}{N} \times \frac{2N^2}{L_S}}{\left(\frac{2(N-C_S)N}{L_S\sqrt{C_S}} + \frac{2C_S - N}{N}\frac{N^2}{L_S\sqrt{C_S}} + \frac{N^2}{L_S\sqrt{C_S}} - \frac{N}{L_S}\sqrt{\frac{2(N-C_S)C_S}{N}}(\frac{N}{C_S}+1)\right)}.$$

Note that $h(N) = \sqrt{C_S}$ if $N = C_S$, and $h(N) = 0$ if $N = 2C_S$.

$2C_S \geq N$. Here there are no subcases, and $B_{opt} = \sqrt{C_S}$.



(a) (b)

Fig. 6: (a) Determining when blocking is useful. (b) Variations of interferences between X and A.

3.3 Threshold Value of N

In this section, the problem of determining the threshold value N_{thr} of N for which blocking is profitable is briefly addressed. Basically, blocking is useful if $G > 0$. According to the theoretical expression of Section 3.1, $G > 0$ if $N > C_S$, i.e., $N_{thr} = C_S$. Now, according to the expressions of Section 3.2.1, if $N < C_S$, $G > 0$ as soon as N is approximately greater than $2\sqrt{C_S}$.

In fact, the theoretical expression only considers capacity misses, so that blocking can only become profitable if X is larger than the cache, i.e., if $N > C_S$. However, cache interferences can occur even when capacity misses still do not occur. As explained in Section 3.2.2, blocking has the effect of reducing the reuse distance of X so that it can be useful for minimizing interferences only. That is why N_{thr} is actually much smaller than C_S. This observation is confirmed by Figure 6(a) ($2\sqrt{C_S} \simeq 64$ for $C_S = 1024$).

3.4 Estimated Precise Gain and Corresponding Optimal Block Size

In Section 2 it has been shown how to obtain a precise evaluation of cross interferences between two arrays. The precise gain is simply obtained by cumulating such expressions for all pairs of arrays, as was done for the average gain in Section 3.2. Because of paper length constraints, the full expressions are not provided here. On the other hand, the differences between average and precise gain are highlighted.

These differences occur when $d = gcd\ (M, C_S)$ is large. The number of possible cache positions of the blocks of size B of A is equal to C_S/d. So, if d is large, there are few such positions. Consequently, the corresponding cache locations are heavily referenced. With respect to X, A appears to be distributed into few cache intervals separated by holes (see Figures 2(a) and 6(b)). Therefore, if array X overlaps with one such interval, interferences between A and X are very intense. Overlapping occurs if the relative cache distance r between A and X is smaller than B (see Figure 6(b), case $r < B$). Overlapping does not occur if $B < r$ and $B + r < d$.

So array base addresses play a significant role when d is large. This is illustrated

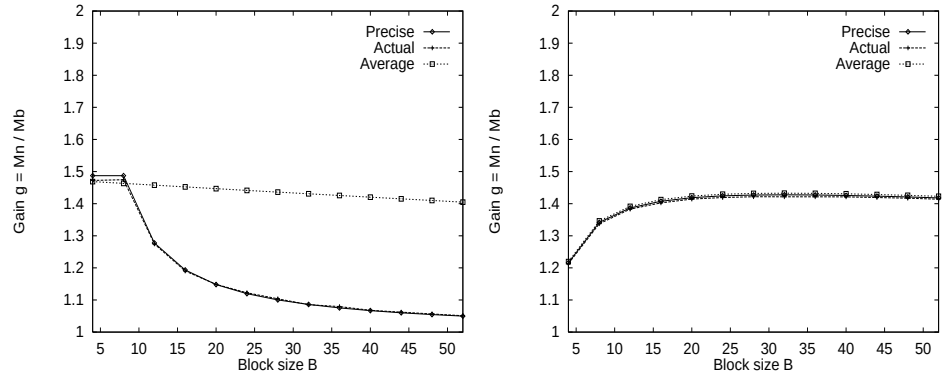


Fig. 7. Influence of array base addresses on the optimal block size ($d = 512$ and $d = 4$).

in Figure 7 where $d = 512$ ($N = 512$), $r = 8$, and B is varied. The actual optimal value of B is equal to 8, which is correctly predicted by the precise estimate. On the other hand, if the average estimate correctly predicts the optimal gain that can be expected, it is independent of r, and therefore it fails to predict for which value of B interferences will occur. For a similar value of N ($N = 516$), $d = 4$, and performance variations are independent of r. Consequently, the average estimate remains precise for all values of B. Also, the average estimate is poor when d is large because array base addresses are not considered. On the other hand, the accurate estimate successfully predicts performance variations.

Note that this particularity of external cross-interferences can be exploited. If there are holes between two intervals of A, then B should be selected small enough that an interval of X fits into one such hole. In this case, no cross-interference occurs between A and X. This cannot be achieved when d is too small (smaller than L_S) because the optimal block size needs to be at least equal to L_S in order to exploit spatial locality.

However, if d is relatively small, it is not obvious that selecting a very small B will yield important benefits considering the tradeoffs imposed by array Y (see Section 3.2.1). Another solution is to adjust the relative base address r (by adjusting the base address of A or X) so that $B < r$. If this is not possible because of potential negative influence on other loops in the code, another solution is simply to copy array X in another array with a suitable base address.

4. APPLICATIONS

The techniques presented at the beginning of this article consist in precisely evaluating the number of external cross-interferences between two arrays by first examining their relative cache positions and then computing the number of overlapping array elements. These techniques have been applied to the three different pairs of arrays that can be found in matrix-vector multiply and which exhibit different patterns of relative cache positions. As far as the loop nest depth is small, these techniques can be extended to any pair of arrays. If the relative position depends on many

indices, the same techniques can be used for the innermost index or indices, and an average estimate can be used for the outer indices (this was done for the pair X, Y).

Accurate evaluation of cache interferences is important for checking whether restructuring techniques do not induce negative side-effects that degrade potential benefits. It clearly appears in this article that blocking is a delicate tradeoff which depends on loop and array parameters.

Including such techniques in a compiler has not yet been achieved, but it is a possible follow-up to this study. A first implementation could be limited to average estimates which can be derived relatively easily. Precise estimates are more difficult to implement, but a first solution is to only *detect* that precise estimates are needed by identifying high-risk cases. For instance, if the relative position of a pair of arrays only depends on one index with coefficient M, the test would be simply to check the value of parameter $d = gcd\ (M, C_S)$. If d is large, intense interferences could occur depending on the array base addresses, and a conservative (but nonoptimal) attitude would be to select a block size of L_S in these cases.

A more immediate application of this model is the development of a linear algebra library finely tuned for caches. Though it is not possible to act on array base addresses, block size adjustment and copying provide sufficient flexibility to exploit fully the different cases detailed in this article.

5. CONCLUSIONS

Several conclusions can be drawn from this analysis of matrix-vector multiply. First, accurately evaluating external cross-interference misses and deriving an analytical expression of the number of such cache misses are tractable tasks. Second, the optimal block size, as computed by current data locality optimizing algorithms, is highly inaccurate, because only capacity misses are considered. If interference misses are taken into account in the optimization problem, the solution then becomes an accurate evaluation of the optimal block size. Third, average estimate of external cross-interferences is frequently but not always sufficient because, in some cases, array base addresses can strongly influence the occurrence and intensity of cache interferences.

REFERENCES

Eisenbeis, C., Jalby, W., Windheiser, D., and Bodin, F. 1990. A strategy for array management in local memory. In *Proceedings of the 3rd Workshop on Programming Languages and Compilers for Parallel Computing*. Irvine, California.

Esseghir, K. 1993. Improving data locality for caches. M.S. thesis, Univ of Texas, Houston, Tex.

Ferrante, J., Sarkar, V., and Thrash, W. 1991. On estimating and enhancing cache effectiveness . In *Proceedings of the 4th Workshop on Languages and Compilers for Parallel Computing*. Santa Clara, California.

Fricker, C., Temam, O., and Jalby, W. 1993. Accurate evaluation of blocked algorithms cache interferences. Tech. rep., Leiden Univ., Leiden, The Netherlands. Mar.

Kane, G. and Heinrich, J. 1992. *MIPS RISC Architecture*. Prentice-Hall, Englewood Cliffs, N.J.

Lam, M., Rothberg, E. E., and Wolf, M. E. 1991. The cache performance of blocked algorithms. In *4th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York , 63-74.

McKinley, K. S. 1992. Automatic and interactive parallelization. Ph. D. thesis, Tech. Rep. CRPC-TR92214, Rice Univ, Houston, Tex.

Porterfield, A. K. 1989. Software Methods for Improvement of Cache Performance on Supercomputer Applications. Ph. D. thesis, Tech. Rep. CRPC-TR89-93, Rice Univ, Houston, Tex.

Sites, R. L. 1992. *Alpha Architecture Reference Manual.* Digital Press, Bedford, Mass.

Temam, O., Fricker, C., and Jalby, W. 1993. Impact of cache interferences on usual numerical dense loop nests. In *Proc. IEEE, special issue on* Computer Performance Evaluation.

Temam, O., Fricker, C., and Jalby, W. 1994. Cache interference phenomena. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems.* (Nashville, Tenn.). ACM, New York.

Wolf, M. and Lam, M. 1991. A data locality optimizing algorithm. In *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation.* SIGPLAN Not. 26, 6, 30-44.

Received January 1994; revised June 1994; accepted February 1995