Alignment of matrices when using XOR-based hashing

Bavo Nootaert*,
Hans Vandierendonck*,
Koen De Bosschere*

* *ELIS, Ghent University, Belgium*

ABSTRACT

To reduce the number of conflicts in caches, hash function are used. One family of hash function that has been reported to perform well in the presence of stride patterns are based on XORing address bits together. We show that, in contrast to traditional modulo-2^m-based hashing, for XOR-based hashing the alignment of a stride pattern has a significant impact. We present some properties that allow to quickly determine an optimal base address.

KEYWORDS: cache, optimization, stride pattern, XOR-based hash function

1 Introduction

Hash functions have been studied with the aim of avoiding conflict misses in caches [SSS93, TG99]. In scientific programs, stride patterns are common. It is important to choose a hash function so as to map as many strides as possible with few conflicts. Conventional bit selection has the disadvantage of mapping even strides onto only one halve of the sets. A well studied alternative are the XOR-based hash functions [FJL85, VDB05], and a special subclass thereof, the polynomial hash functions [Rau91]. These functions are computed using arithmetic over GF(2) and can be evaluated using solely XOR-gates.

In [TG99] benchmarks are used to measure the performance of these hash functions. Analytical studies of XOR-based hash functions are usually limited to vector space patterns[FJL85, VDB05]. Analyzing these patterns is rather restrictive, since this class contains only certain strides, aligned to certain base addresses. In [Rau91], some properties are proven about the effect of polynomial hash functions on stride patterns starting at base address 0. Although some of the theorems can be extended to include a different base address [NVDB05], they are aimed at determining equivalence among patterns, and generally cannot say whether a particular base address is actually a good choice. Indeed, the number of conflicts varies greatly depending on the alignment of the stride pattern. This will be shown in the Section 2. Using the model outlined in Section 3, we will give some properties that allow to refine the search for the optimal base address to a limited range (Section 4).

```
for( k=0; k < S; ++k )
  C[i,j] += A[i,k] * B[k,j]
```

Figure 1: Example loop

2 An example

The code fragment in Figure 1 is taken from matrix multiplication. In this loop, there is one pattern with stride S, and one with stride 1. Eventually, every row of matrix A and every column of matrix B will be accessed, i.e. there are S patterns with stride S and S with stride 1. Let a be the starting address of matrix A. The patterns with stride S start at addresses a, $a+1$, ..., $a+S-1$, and the others start at addresses a, $a+S$, ..., $a+(S-1)S$. We want to align the matrices so that self-interference [LREW91] of the rows and the columns is avoided as much as possible when mapped onto the cache. In this paper, we will restrict ourselves to just one matrix and assume that the matrices do not cross-interfere. This assumption is acceptable when the cache is 2-way set-associative, or the dimension of the matrix is small compared to the number of sets.

Consider a cache with 2^{13} sets and a mapping that XORs two slices of 13 bits together, and discards any bits above the 26 least significant. Assume for simplicity that a cache line can hold exactly one element of the matrix.

Figure 2 shows the number of conflicts for $S = 976$ and base addresses ranging from 0 to $5,000,000$. Here, optimal placement reduces the number of conflicts with about 59% compared to the worst case. As can be observed, there is some symmetry. This is the property used in Section 4.

3 Mathematical model of the placement problem

XOR-based hash functions that map an n-bits address to an m-bits set index are defined by an $n \times m$ matrix H over GF(2). An address q is interpreted as a bit vector and is mapped to the index qH. Matrix multiplication is defined as usual, but all operations are performed modulo 2. So addition and multiplication are equal to XOR and AND, respectively.

A *conflict* occurs when an element of a pattern is mapped to a set that already contains an element. The number of conflicts is independent of the order in which the elements are mapped. This allows for a study of the performance of a hash function that is independent of any particular benchmark. We define the number of *conflicts of a matrix* as the sum of the conflicts of the column access patterns and the row access patterns. Two patterns X_1 and X_2 are called *equivalent* for a hash function H if they show the same number of conflicts. This is denoted by $X_1 \equiv_H X_2$.

4 Exploiting the symmetry

The number of conflicts shows many symmetries as the base address of the matrix varies (Figure 2). We characterize these symmetries and exploit them to pinpoint regions of base addresses that may contain optimal base addresses, i.e., base addresses that minimize the number of conflicts. Once an optimal base address is found in such a region, other optimal

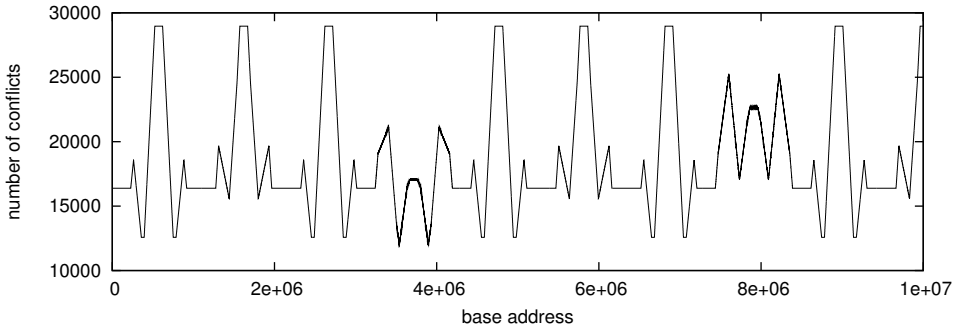

Figure 2: The number of conflicts for the example loop

base addresses may be found in different regions by exploiting the symmetries. Proofs can be found in [NVDB05].

4.1 Single stride patterns

For a pattern with stride S of size W, the symmetry points are given by:

$$b_j = 2^{j+t} + \left\lceil \frac{S}{2} \right\rceil - \left\lceil \frac{WS}{2} \right\rceil,$$

where t is the smallest integer that makes b_0 positive. I.e. for all $\beta \in [0, b_j]$ and all XOR-based hash functions H,

$$\begin{array}{lll} \{Si + b_j - \beta | 0 \leq i < W\} & \equiv_H \{Si + b_j + \beta + 1 | 0 \leq i < W\} & \text{if } S \text{ is even} \\ \{Si + b_j - \beta | 0 \leq i < W\} & \equiv_H \{Si + b_j + \beta | 0 \leq i < W\} & \text{if } S \text{ is odd} \end{array}$$

The range $[0, 2b_j]$ does not reach b_{j+1}. So if one examines the base addresses starting from 0, the range $]2b_j, b_{j+1}]$ contains new information. In this range, there may be an optimal base address that has not yet been discovered.

After the range $[0, 2b_0]$, which is symmetric since b_0 is its center, has been reflected around b_1, both the original and its image are reflected again around b_2, resulting in four copies of the range. After the next symmetry point, we have eight copies, and so on.

To find a base address for which the pattern is optimally mapped, one only needs to search through the base addresses in the *primitive ranges* $]2b_j, b_{j+1}]$, i.e. the ranges containing the base addresses for which the mapping cannot be computed from the previous base addresses. The width of each primitive range is $\delta = \lceil WS/2 \rceil - \lceil S/2 \rceil$.

Since $b_j + \delta = 2^{j+t}$, there are $n - t + 1$ ranges over which the number of conflicts need to be computed.

4.2 Composite stride patterns

A composite stride pattern is an ordered collection of stride patterns with the same stride S and such that the distance between the base addresses of neighboring patterns is a fixed number d. E.g. the columns of an $S \times S$ matrix stored in row major order are a composite stride pattern with $d = 1$, and the rows are a composite stride pattern with $d = S$. The symmetry points for a composite stride pattern, consisting of R individual patterns, are:

$$b'_j = 2^{j+t'} + \left\lceil \frac{S}{2} \right\rceil - \left\lceil \frac{WS}{2} \right\rceil - \left\lceil \frac{Rd}{2} \right\rceil - 1,$$

where t' is again the smallest integer that makes b'_0 positive.

4.3 Matrix patterns

It can be proven that the symmetry points for the column pattern and the row pattern of a square matrix coincide.

Returning to the example given in Section 2, we have $b'_0 = 48000$, $b'_1 = 572288$, $b'_2 = 1620864$, ... Optimal base addresses are any address in the range $[3539056, 3539296]$, and all of its equivalents obtained by reflection around the symmetry points.

5 Conclusion

The base address of a stride pattern or a matrix pattern has a significant impact on the number of conflicts. By carefully placing the pattern in memory, the number of conflicts can be reduced. We have given some properties by which the optimal base address can be calculated in linear time for fixed stride and pattern size.

References

[FJL85] J. M. Frailong, W. Jalby, and J. Lenfant. XOR-schemes: a flexible data organization in parallel memories. In *Proceedings of the 1985 International Conference on Parallel Processing*, pages 276–283, August 1985.

[LREW91] M. D. Lam, E. E. Rothberg, and M. E. E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 63–74, April 1991.

[NVDB05] B. Nootaert, H. Vandierendonck, and K. De Bosschere. How patterns in memory references affect the performance of hash functions in cache memories. Technical Report R105.002, Ghent University, March 2005.

[Rau91] B. R. Rau. Pseudo-randomly interleaved memory. In *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pages 74–83, May 1991.

[SSS93] M. Schlansker, R. Shaw, and S. Sivaramakrishnan. Randomization and associativity in the design of placement-insensitive caches. Technical report, HP Computer Systems Laboratory, June 1993.

[TG99] N. Topham and A. González. Randomized cache placement for eliminating conflicts. *IEEE Transactions on Computers*, 48(2):185–192, 1999.

[VDB05] H. Vandierendonck and K. De Bosschere. XOR-based hash functions. *IEEE Transactions on Computers*, 54(7):800–812, 2005.