From the fractal dimension of the intermiss gaps to the cache-miss ratio

by Dominique Thiébaut

This work extends a model proposed by Voldman, Mandelbrot, et al. on the fractal nature of the gaps separating cache misses, and shows how the fractal dimension of the gap distribution can be used to predict the miss ratio experienced by the program that has generated the series of cache misses. This result supports the thesis that the fractal dimension of the distribution of the intermiss gaps is a potentially powerful measure for program characterization.

1. Introduction

Voldman, Mandelbrot, et al. have studied the activity of several programs in memory caches, and have shown that the seemingly erratic occurrences of gaps separating two consecutive cache misses very closely fit the model of fractal elements. Mandelbrot first introduced and then developed a complete theory of fractal geometry with an astonishing number of applications to natural and physical phenomena [2]. In particular, if we define the gap between two cache misses (referred to as the *intermiss gap* in the remainder of this paper) as the number of consecutive cache hits occurring between two cache misses plus one, and if we call G the random variable whose values are the successive gaps experienced by some program P, then, over the whole execution of program P, the probability distribution $\Pr[G > u]$ is hyperbolic and closely follows a curve of equation

$$\Pr[G > u] = Au^{-\theta}, \tag{1}$$

where θ is introduced by Mandelbrot as the *fractal dimension*. This property is very easily recognizable when the distribution is plotted with a doubly logarithmic scale, as the curve then becomes linear with slope θ.

In this paper, we verify that programs running on computers of a much smaller size than that of the computer used by Voldman and Mandelbrot for their analysis also exhibit a memory access pattern giving rise to a fractal intermiss-gap distribution, and that we can evaluate the cache-miss of a program from certain particular knowledge of the fractal dimension θ of its intermiss-gap pattern. This evaluation, however, is not exact. It is based on the assumption that the intermiss-gap distribution follows Equation (1). As a result the approximation is directly related to the degree with which the intermiss-gap distribution *fits* the fractal model.

Others have attempted to predict miss ratios by different means. Laha, Patel, and Iyer [3] propose a statistically based scheme in which sampling techniques are applied while the address trace of the workload under study is being generated. Agarwal, Horowitz, and Hennessy [4] propose a hybrid method based on both trace analysis and analytical modeling for the prediction of the miss ratio of a given workload.

Strecker [5] extends work by Easton and Fagin on the cold-start miss ratio [6], and proposes a cache-reload transient model for the prediction of the miss ratio of interleaved programs based on the continuous-flow model. This approach requires the analysis of a trace of the workload under consideration for different cache sizes, from which several parameters representative of the workload are extracted, allowing for the prediction of the miss ratio for cache sizes other than those studied.

All of these authors base their computation of the miss ratio on an analysis of the trace representing the workload under investigation. Laha et al. rely on sampling techniques, while the other authors generally filter the whole trace. In their analysis they try to describe, through the use of several parameters, the erratic dynamics characterizing the behavior of programs and, indirectly, the associated miss ratios.

Our contribution in this paper is an attempt to predict the miss ratio with only one parameter, also derived from an analysis of a program trace but carrying what we believe is a very powerful measure of program behavior. This approach requires that a trace of a given program be fed to a cache simulator. Instead of recording the number of misses as a function of the total number of cache accesses, we record the gaps between consecutive misses. Hence, the goal of this paper is not so much to present a more efficient methodology for computing the miss ratio of a given trace in a fixed-size cache. It is rather to further stress the extreme richness of the fractal parameter in information about the behavior of the program, and in particular, the possibility of predicting the cache-miss ratio of the program from knowledge of the fractal dimension of the intermiss gaps. This ability of the fractal parameter to model program behavior in memories has been reported recently in [7], where it is shown that, when modeled as a fractal random walk through the memory lattice, the memory access pattern of computer programs exhibits a fractal dimension, and that knowledge of this alone can be used to predict the miss ratio of the programs in fully associative caches of any size.

In the next section we derive an approximation of the cache-miss ratio from the knowledge of the intermiss-gap distribution. In the third section we apply this result to traces of computer programs and show the relationship between the real miss ratio and the one computed using the fractal model, when one or two fractal dimensions are introduced. The fourth section concludes this paper.

2. Derivation of the cache-miss ratio

To show how the miss ratio is related to the plot of the intermiss-gap distribution $\Pr[G > u]$, we show by example how such a plot is derived, and then how the components of the miss ratio can easily be computed.

Let us assume that we have recorded the series of gaps separating cache misses during the execution of program P, and further that this record is the realization of a random variable G. We denote this series of events as $\{G\}$. From $\{G\}$ we generate a sorted list $\{G_i, N_i\}$ of the gaps, such that the gaps are sorted in ascending order by their length G_i over the range $i = 1, \cdots, p$, and such that each unique length G_i is associated with a number N_i representing the total number of gaps of length G_i in the series $\{G\}$. The following equation is always true:

$$\sum_{i=1}^{p} N_i = \text{Card}(\{G\}) = N,$$

where $\text{Card}(\cdot)$ represents the total number of elements in the series $\{G\}$. It is straightforward to derive the probability distribution $\Pr[G > G_i]$, where G is the random variable and G_i a realization of G:

$$\begin{aligned}\Pr[G > G_i] &= \Pr[G = G_i] \\ &\quad + \Pr[G = G_{i+1}] + \cdots + \Pr[G = G_p] \\ &= \frac{N_i}{N} + \frac{N_{i+1}}{N} + \cdots + \frac{N_p}{N} \\ &= \frac{\sum_{j=i}^{p} N_j}{\sum_{j=1}^{p} N_j},\end{aligned}$$

where p is the index of the longest gap(s) in the sorted list $\{G_i, N_i\}$. Since we denote the smallest gap as G_1, p is also the number of unique gaps in the series $\{G\}$:

$$p = \text{Card}(\{G_i, N_i\}).$$

Assuming that the series of intermiss gaps indeed exhibits a fractal behavior, the probability distribution $\Pr[G > G_i]$ is hyperbolic, and can be written as

$$\begin{aligned}\Pr[G > G_i] &= \frac{\sum_{j=i}^{p} N_j}{\sum_{j=1}^{p} N_j} \\ &= A' G_i^{-\theta}, \qquad (2)\end{aligned}$$

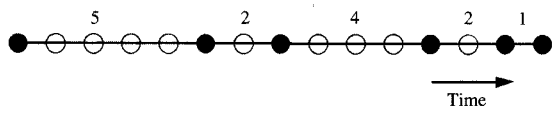
where A' and θ are positive real numbers. Since the denominator of Equation (2) is constant for any given series $\{G\}$, we can, without any loss of generality, shift from the probabilities to the distribution of the N_i,

$$\sum_{j=i}^{p} N_j = A G_i^{-\theta}. \qquad (3)$$

The cache-miss ratio associated with the series $\{G\}$ is defined as follows:

$$\textit{miss ratio} = \frac{\textit{number of misses}}{\textit{number of misses} + \textit{number of hits}}.$$

Since each intermiss gap is measured between two cache misses, the total number of misses is simply equal to the total number of gaps in the series $\{G\}$ plus one. This simple relationship is illustrated in **Figure 1**.



● Cache miss ○ Cache hit

$$\{G\} = \{5, 2, 4, 2, 1\}$$

$$\{G_i, N_i\} = \{(1, 1), (2, 2), (4, 1), (5, 1)\}$$

$$number\ of\ gaps = 5$$

$$number\ of\ misses = 6$$

$$number\ of\ cache\ accesses = (1 \times 1) + (2 \times 2) + (4 \times 1) + (5 \times 1) + 1 = 15$$

Figure 1

A series of five gaps.

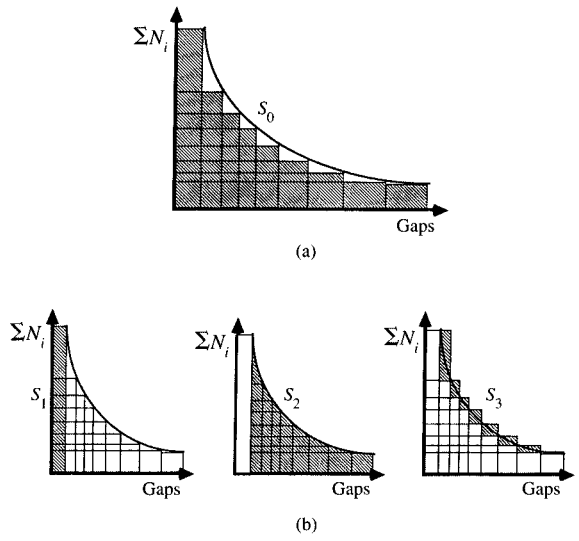


Figure 2

Computation of the area S_0.

As a result, we have the following two equations:

$$number\ of\ misses = 1 + \sum_{i=1}^{p} N_i \tag{4}$$

and

$$number\ of\ accesses = 1 + \sum_{i=1}^{p} N_i G_i. \tag{5}$$

From Equations (3) and (4) we deduce a simple expression for the number of misses in terms of A, θ, and the length of the shortest gap recorded:

$$number\ of\ misses = 1 + AG_1^{-\theta}. \tag{6}$$

The second term of Equation (5) is simply the area of the rectangles under the hyperbolic curve defined by A and θ, as shown in **Figure 2(a)**. (Note: We abandon the log/log scale in Figure 2, since we are interested in the area under the hyperbolic curve.) We refer to this area as S_0. We compute an approximation of S_0 from the areas S_1, S_2, and S_3, shown in **Figure 2(b)**, in the following manner:

$$S_0 = \sum_{i=1}^{p} N_i G_i$$

$$\simeq S_1 + S_2 - \frac{S_3}{2}.$$

S_1 and S_2 are given exactly by

$$S_1 = G_1 \sum_{i=1}^{p} N_i$$

$$= G_1 A G_1^{-\theta}$$

$$= AG_1^{1-\theta}; \tag{7}$$

$$S_2 = \int_{G_1}^{G_p} Ag^{-\theta}\, dg$$

$$= \frac{A}{1-\theta}(G_p^{1-\theta} - G_1^{1-\theta}); \tag{8}$$

and S_3 is approximated by the following quantity:

$$S_3 = \sum_{i=1}^{p-1} N_i(G_{i+1} - G_i)$$

$$\simeq \sum_{i=1}^{p-1} N_i \quad \text{(assuming that } G_{i+1} - G_i = 1 \text{ for any } i)$$

$$= -N_p + \sum_{i=1}^{p} N_i$$

$$= -AG_p^{-\theta} + AG_1^{-\theta}.$$

In combination, Equations (7), (8), and (9) yield

$$S_0 \simeq AG_1^{1-\theta} + \frac{A}{1-\theta}(G_p^{1-\theta} - G_1^{1-\theta}) - \frac{1}{2}A(G_1^{-\theta} - G_p^{-\theta}).$$

Combining this result with Equation (5), we get the total number of cache accesses,

$$\textit{number of accesses} = 1 + A\left(\frac{G_1^{-\theta}}{2} - \frac{\theta G_1^{1-\theta}}{1-\theta} + \frac{G_p^{1-\theta}}{1-\theta} + \frac{G_p^{-\theta}}{2}\right). \quad (10)$$

Finally, the miss ratio is computed as the ratio of Equation (6) to Equation (10):

$$\textit{miss ratio} = \frac{1 + AG_1^{-\theta}}{1 + A\left(\frac{G_1^{-\theta}}{2} - \frac{\theta G_1^{1-\theta}}{1-\theta} + \frac{G_p^{1-\theta}}{1-\theta} + \frac{G_p^{-\theta}}{2}\right)}, \quad (11)$$

which we can further approximate as

$$\textit{miss ratio} \simeq \frac{G_1^{-\theta}}{\left(\frac{G_1^{-\theta}}{2} - \frac{\theta G_1^{1-\theta}}{1-\theta} + \frac{G_p^{1-\theta}}{1-\theta} + \frac{G_p^{-\theta}}{2}\right)}. \quad (12)$$

This final equation is interesting because it does not depend (in our approximation) on the constant A, but on θ, G_1, and G_p. Since the smallest gap recorded will most probably be equal to 1, Equation (12) can be simplified by replacing G_1 with 1. We can further simplify it by assuming that the longest gap recorded, G_p, will occur only once in the series $\{G\}$, in which case $N_p = 1$ and G_p is given directly by Equation (3),

$$AG_p^{-\theta} \simeq N_p = 1, \quad (13)$$

or

$$G_p = A^{1/\theta}. \quad (14)$$

In this case the miss-ratio expression can take the following simple form:

$$\textit{miss ratio} \simeq \frac{2}{\frac{1-3\theta}{1-\theta} + G_p^{-\theta}\left(\frac{2G_p}{1-\theta} + 1\right)}. \quad (15)$$

3. Experimental verification

We tested the validity of Equation (12) by plotting the intermiss-gap distribution of several programs and by applying a least-squares fit to obtain the values of A and θ best representing the empirical data. The analysis is based on *traces* of the execution of several programs. The traces contain the addresses of consecutive memory accesses generated by different programs while fetching both *data* and *code* information stored in memory. The tracer that we developed for this purpose captures all memory accesses generated by the programs' instructions, with the exception of instructions imbedded in critical sections (uninterruptible code). The programs were captured on an Intel 8088/8086-based computer, with 512K bytes of main memory and nonvirtual memory management. The programming was a uniprogramming environment. It included a Pascal compiler (Turbo Pascal®[1], Version 2), a Basic interpreter (Microsoft®[1] Basica), a spell checker (IBM WordProof™[1]), and a text editor (Turbo Pascal editor) executing a string search. The cache simulated is a 128-congruence-class, eight-way-associative cache, with a line size of eight words. Each congruence class is managed by a least recently used (LRU) algorithm. The selection of the congruence class is performed by taking the modulo of the line address with respect to the number of congruence classes.

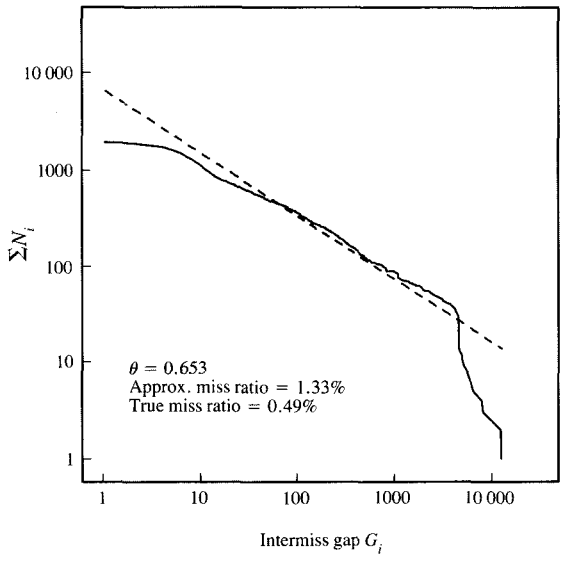


Figure 3

Basic interpreter: Intermiss-gap distribution (solid curve, true distribution; dashed curve, regression).

The plots of the intermiss-gap distributions along with the hyperbolic curves obtained by linear regression are shown in **Figures 3–6**. The miss ratios obtained and the relative error of prediction are summarized in **Table 1**.

The difference between the predicted and true miss ratios stems from several factors. The first, most obvious factor is that all four programs tested show a "knee" in their intermiss-gap distribution. Such a phenomenon also appears in the study by Voldman and Mandelbrot, and they show that the knee corresponds to gaps equal to half the size of the cache. We refer to this knee as the cache-size knee. The second factor in the observed difference is linked to the regression algorithm. Because the regression algorithm is applied to the *totality* of the points defining our

[1] Turbo Pascal is a registered trademark of Borland International, Inc., Scotts Valley, CA. Microsoft is a registered trademark of Microsoft Corporation, Seattle, WA. WordProof is a trademark of International Business Machines Corporation.

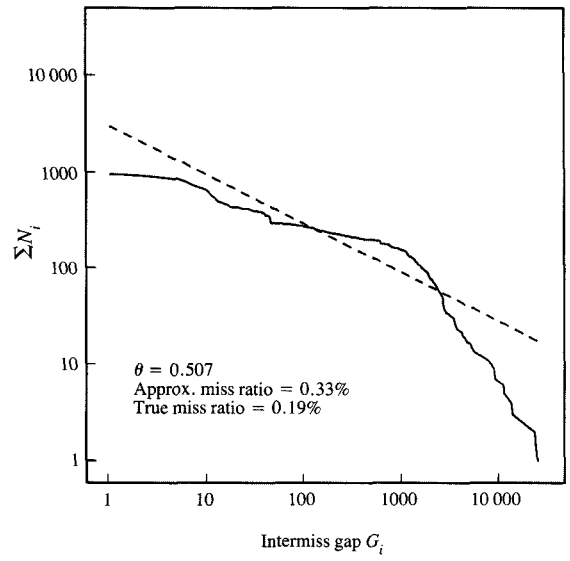


Figure 4

Spell-checker: Intermiss-gap distribution (solid curve, true distribution; dashed curve, regression).

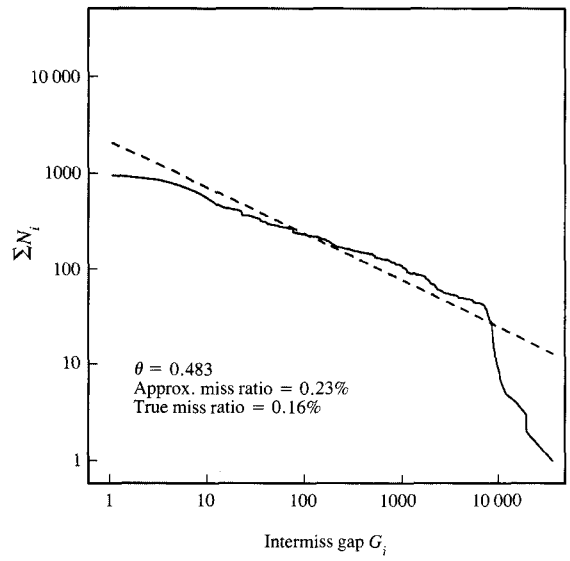


Figure 5

Pascal compiler: Intermiss-gap distribution (solid curve, true distribution; dashed curve, regression).

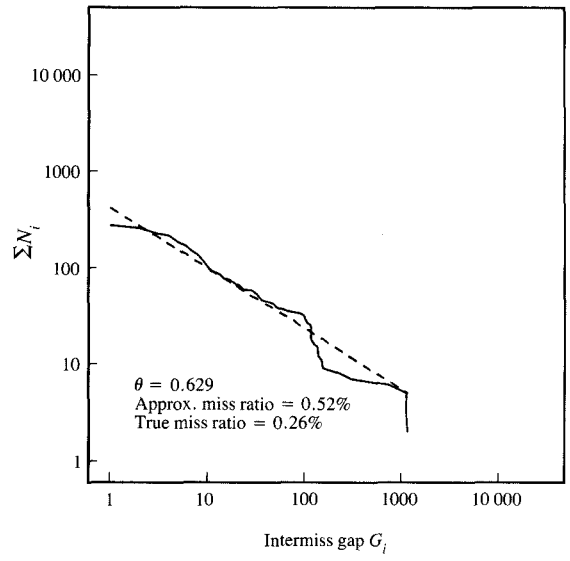


Figure 6

String search program: Intermiss-gap distribution (solid curve, true distribution; dashed curve, regression).

experimental distribution, the hyperbolic curve never completely fits the flat portion of the experimental distribution. Instead, its slope is always steeper than the longest, flattest portion of the experimental curve. Voldman and Mandelbrot similarly point out that the determination of the slope θ of the experimental curve is critical, but they do not detail their algorithm for its computation.

The curves also show a second, much smoother, knee around the gap value of eight. Even though the knee is slight, its influence is critical, because it is located at the high end of the ordinate logarithmic scale, and any variation from the linear interpolation curve in this area translates into large numbers. This in turn translates into a significant error in our approximation of the miss ratio. We conjecture that the location of this knee is related to the size of the cache lines, and we refer to this knee as the line-size knee. It appears that this soft knee is more accentuated when the cache has a low degree of associativity.

The influence of the line size on the number of misses, and in our case, on the gaps between misses, can be explained in the following way. Assume that a program accesses, in a completely random fashion, all 16 memory items located from address 100 to 10F (hexadecimal). If we assume a line size of one word, and that none of the items are in the cache to start with, the 16 accesses result in 16 misses separated by a gap of length one. If we now assume

that the line has a size of eight words, then the same 16 accesses will produce only two misses, one miss when the first item in the range 100 to 107 is accessed, the other miss when the first item in the range of 108 to 10F is accessed. Some of the information relating to the access pattern is thus "masked" by the line size. This information, were it known, would contribute to the frequency of occurrence of gaps of length one to seven. Instead, a gap of length between one and eight[2] is recorded. Therefore, we can expect the frequency counters associated with the gaps of lengths one to seven to be the ones most affected by the masking introduced by the line size. To compensate for this line-size effect, we conducted two experiments.

In the first experiment, we recorded only gaps of length greater than or equal to eight. As the program trace was fed to caches of different sizes, only those gaps between misses separated by at least seven memory accesses contributed to the total number of misses and to the individual frequency counters associated with each possible gap length. We refer to this experiment as the *truncation* experiment.

In the second experiment, we recorded all gaps, including gaps of length less than eight, but this time we did not maintain individual frequency counters for these gaps. Instead, we maintained a single counter for all of them. We chose to associate that counter with a gap length of eight. A short example will clarify this concept. Assume that frequency counters for gaps of length 8 to some large number are all initialized to 0. Assume furthermore that when the trace of a given program is fed to the cache, we observe misses separated by the following gaps: {1, 1, 2, 1, 9, 3, 5, 8, 9, 15, · · · }. After the gap of length 15 is analyzed, the counter of the total number of misses contains 10; the frequency counters associated with gaps of length 8, 9, and 15 contain 7, 2, and 1, respectively. We have, in essence, *collapsed* all the gaps of lengths 1 to 8 into one category, and decided that its length would be 8.

In both experiments, the first step consists in applying a regression algorithm to compute two values of θ, respectively called θ_1 and θ_2. θ_1 represents the slope of the intermiss-gap distribution on the left side of the cache-size knee, while θ_2 represents the slope of the distribution on the right side of the same knee. The location of the knee varies with every cache simulated. As Voldman has shown, this location is usually in the vicinity of twice the size of the cache considered. Our algorithm for finding θ_1 and θ_2 can be summarized as follows:

- Select a cache of size C lines, where one line contains eight memory words, and where the number of congruence classes is selected so that the degree of associativity remains independent of the cache size, at a value of eight lines per congruence class.
- Feed the trace of the program considered to the software cache-simulator and record the gap between consecutive misses.
- Construct the distribution of the gaps in terms of the quantity $N(u)$, defined as

  $N(u) = Number[recorded\ gaps > u]$.

- Set the cache-size knee at the gap value equal to $C/4$ (since each line contains eight words, half the size of the cache expressed in words is $C \cdot 8/4$).
- Take the logarithm of both components of the (x, y) pairs associated with each point of the distribution $N(u)$, and apply a linear regression algorithm to the part of the curve located on the left side of the cache-size knee. The second regression coefficient yields the value of θ_1. Similarly, the

[2] The gap in this case can only be of maximum length eight, if we assume that the first eight items accessed are all part of the same line.

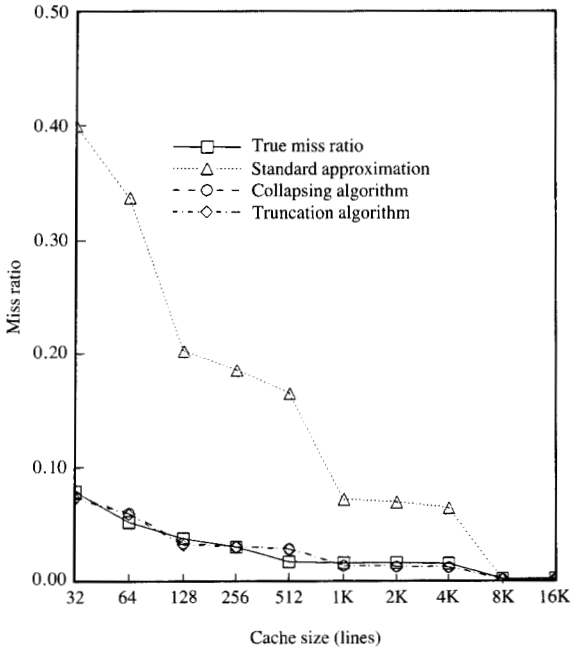


Figure 7 Influence of line size on prediction of miss ratio.

Table 1 True and approximated miss ratios.

Program	*Pascal compiler*	*Basic interpreter*	*Spell checker*	*Text editor*
True miss ratio (%)	0.23	0.49	0.19	0.26
Approx. miss ratio (%)	0.16	1.33	0.33	0.52
Relative error (%)	30	171	73	100

Table 2 Miss ratio and fractal dimensions. Linear interpolation applied on both sides of cache-size knee (NA = Not Applicable).

Cache size (lines)	*32*	*64*	*128*	*256*	*512*	*1024*	*2048*	*4096*	*8192*	*16384*
θ_1	2.00	1.71	1.30	1.27	1.20	0.96	0.96	0.95	0.47	0.48
θ_2	1.32	1.06	1.10	0.97	0.84	1.06	0.99	1.48	NA	NA
True miss ratio (%)	7.9	5.2	3.8	3.0	1.7	1.6	1.6	1.5	0.2	0.2
Approx. miss ratio (%)	39.9	33.7	20.2	18.5	16.4	7.2	6.9	6.4	0.2	0.2
Absolute error (%)	32.0	28.6	16.4	15.6	14.8	5.5	5.3	4.9	0.05	0.01

Table 3 Miss ratio and fractal dimensions. Linear interpolation applied on both sides of cache-size knee. Gaps of length 1, 2, ···, 8 collapsed into one category (NA = Not Applicable).

Cache size (lines)	*32*	*64*	*128*	*256*	*512*	*1024*	*2048*	*4096*	*8192*	*16384*
θ_1	2.63	2.12	1.43	1.39	1.31	1.00	1.00	0.99	0.49	0.50
θ_2	1.32	1.06	1.10	0.97	0.84	1.06	0.99	1.48	NA	NA
True miss ratio (%)	7.9	5.2	3.8	3.0	1.7	1.6	1.6	1.5	0.2	0.2
Approx. miss ratio (%)	7.4	6.0	3.3	3.0	2.8	1.4	1.3	1.2	0.1	0.1
Absolute error (%)	0.5	0.8	0.5	0.0	1.2	0.3	0.3	0.3	0.1	0.1

Table 4 Miss ratio and fractal dimensions. Linear interpolation applied on both sides of cache-size knee. Gaps of length 1, 2, ···, 7 removed from distribution (NA = Not Applicable).

Cache size (lines)	*32*	*64*	*128*	*256*	*512*	*1024*	*2048*	*4096*	*8192*	*16384*
θ_1	2.61	2.11	1.42	1.39	1.31	1.00	1.00	0.99	0.49	0.50
θ_2	1.32	1.06	1.10	0.97	0.84	1.06	0.99	1.48	NA	NA
True miss ratio (%)	7.9	5.2	3.8	3.0	1.7	1.6	1.6	1.5	0.2	0.2
Approx. miss ratio (%)	7.3	5.9	3.2	3.0	2.8	1.4	1.3	1.2	0.1	0.1
Absolute error (%)	0.6	0.8	0.6	0.0	1.1	0.3	0.3	0.3	0.1	0.1

same analysis on the right side of the cache-size knee yields the value of θ_2.

The results of our experiments appear in **Tables 2**, **3**, and **4**, and are graphically illustrated in **Figure 7.** Table 2 shows the result of the interpolation and algorithms before application of either the truncation or the collapsing method. Table 3 shows the effect of collapsing the gaps of length eight or less into one bin. Notice that the absolute error in the miss ratio for small cache sizes decreases dramatically. Table 4 shows the effect of not recording the gaps of length one to seven. Here again, the miss-ratio error is greatly reduced. The collapsing method shows a slight advantage over the truncation method in miss-ratio accuracy.

The first major observation we gather from these tables is that, for caches with a small number of congruence classes, the line-size knee creates a considerable error in the approximation of the miss ratio. In this light, we can see that the size of the cache, and in particular the number of congruence classes, influences the "clarity" with which the fractal picture is rendered, very much like the focus of a lens. At the extreme ends of the range of cache sizes, the misses will be occurring either too often or only at start-up, blurring the observed fractal behavior. For large numbers of congruence classes, the fractal behavior is more visible, because the intermiss gaps more closely exhibit a hyperbolic behavior and show a smoother line-size knee than at smaller numbers of congruence classes.

It is surprising that such a large error occurs, given that the intermiss-gap distribution is fitted with two hyperbolic curves with different fractal-dimension coefficients, an operation which greatly increases the accuracy of our algorithm. The second observation is that removing the line-size knee by way of collapsing or deleting the gaps of length less than or equal to eight yields very good approximations of the miss ratio for the different cache sizes studied. In both cases, the maximum absolute error is 1.15%, with a slight advantage in accuracy for the collapsing method.

We conjecture that both the small number of classes and the prefetching introduced by a line size greater than one degrade the observed hyperbolic characteristics of the intermiss-gap distribution for gaps of length less than the line size. By collapsing the gaps of length less than the line size, we can *accentuate* the hyperbolic behavior of the gap distribution, resulting in a significantly enhanced prediction of the miss ratios.

4. Conclusion

In this paper we have verified that smaller programs than those tested by Voldman and Mandelbrot also present a fractal nature in their generation of cache misses. This strongly suggests that the fractal nature of the intermiss gaps is an integral part of the access pattern of computer programs, and is independent of the size of the computer considered. Mandelbrot showed that this fractal characteristic can be parameterized by the fractal dimension θ. We have shown that the knowledge of θ alone is sufficient to approximate the cache-miss ratio associated with the series of intermiss gaps. This last operation is, however, not exact and its accuracy depends largely on the degree to which the distribution fits the hyperbolic model. We show that if we introduce two fractal dimensions, θ_1 describing the fractal behavior of the small gaps and θ_2 describing the fractal behavior of the larger gaps, with a threshold dependent only on the cache architecture, and if we compensate for the influence of prefetching due to a line size greater than one, we can derive a much more precise estimate of the miss ratio. This argument reinforces our belief that the fractal nature of the intermiss gap is a very powerful characterization of program behavior, and that most of the parameters traditionally used to measure program performances are imbedded in the fractal model.

Acknowledgments

We wish to thank Jean Voldman of the IBM T. J. Watson Research Center for enlightening discussions, and Harold S. Stone, also at IBM, for his helpful comments and guidance as well as his critical review of the first draft of this document. We also would like to acknowledge the anonymous reviewers for their constructive comments. This work was supported by International Business Machines Corporation under Grant No. 5-20947.

References

1. J. Voldman, B. B. Mandelbrot, L. W. Hoevel, J. Knight, and P. Rosenfeld, "Fractal Nature of Software-Cache Interaction," *IBM J. Res. Develop.* **27,** No. 2, 164–170 (March 1973).
2. B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman & Co., San Francisco, 1982.
3. S. Laha, J. H. Patel, and R. K. Iyer, "Accurate Low-Cost Methods for Performance Evaluation of Cache Memory Systems," *Technical Report*, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, Urbana, IL, 1986, pp. 1–26; accepted for publication by *IEEE Trans. Comput.*
4. A. Agarwal, M. Horowitz, and J. Hennessy, "An Analytical Cache Model," *Technical Report CSL-TR-86-304*, Stanford University, Stanford, CA, September 1986, pp. 1–45.
5. W. D. Strecker, "Transient Behavior of Cache Memories," *ACM Trans. Comput. Syst.* **1,** No. 4, 281–293 (November 1983).
6. M. C. Easton and R. Fagin, "Cold-Start Versus Warm-Start Miss-Ratios," *Commun. ACM* **21,** No. 10, 866–871 (October 1978).
7. D. Thiébaut, "Influence of Program Transients in Computer Cache-Memories," Ph.D. Thesis, University of Massachusetts, Amherst, MA, February 1988.

Received November 2, 1987; accepted for publication July 25, 1988

Dominique Thiébaut *Department of Computer Science, Smith College, Northampton, Massachusetts 01063.* Dr. Thiébaut received his Bachelor's degree in computer science from the Institut de Programmation, Université de Paris VI, in 1980, and his Master's degree and Ph.D. in electrical and computer engineering from the University of Massachusetts at Amherst in 1983 and 1988, respectively. From 1982 to 1984, Dr. Thiébaut worked at Assurance Technology Corporation, where he designed a computing system for a space satellite. He is currently an Assistant Professor in the Department of Computer Science at Smith College, Northampton, Massachusetts.