J. Voldman
B. Mandelbrot
L. W. Hoevel
J. Knight
P. Rosenfeld

Fractal Nature of Software-Cache Interaction

This paper uses fractals to model the clustering of cache misses. The clustering of cache misses can be quantified by a single number analog to a fractional dimension, and we are intrigued by the possibility that this number can be used as a measure of software complexity. The essential intuition is that cache misses are a direct reflection of changes in locality of reference, and that complex software requires more frequent (and larger) changes in this locality than simple software. The cluster dimension provides a measure (and perhaps the basis for a model) of the intrinsic differences between workloads. In this paper, we focus on cache miss activity as a discriminate between interactive and batch environments.

Introduction

In an earlier paper, we established that cache misses occur in "bursts" [1]. By burst, we mean a clustering of misses over some measure of time. When the intermiss distance is large, we call the intermiss interval a "gap." A burst is followed in time by a gap, forming a primitive *burst-gap* pair. Intuitively, a burst denotes a period of time in which a (software) process develops a working set, while a gap represents an interval in which the process computes within the working set established by the burst.

This paper proposes a mathematical model relating bursts to a programmer's view of software structure. The model implies a definition of *reference locality* that does not require the specification of an *a priori*, fixed-size time window to compute working sets and cache miss ratios. We view each burst-gap pair as defining a natural working set (of cache lines) gathered during the burst and used during the gap.

Fractals

Fractal geometry is a mathematical theory conceived and developed to provide a model for irregularity and fragmentation in nature (Ref. [2]). It is beyond the scope of this paper to present or summarize this theory. Instead, we attempt in this section only to convey an intuitive justification for its application to the study of software-cache interaction.

As defined by Denning in Ref. [3], the notion of reference locality requires one to select a time window within which unique memory references are counted. The set of unique references within a given window is called the "*working set over the defined window.*" Unfortunately, by varying the window size, one ends up with different working sets. In other words, what is in the window depends on the window size.

This reminds us of the arbitrariness encountered in estimating the length of a coastline using an arbitrary measuring unit, as described in Ref. [2]. Each measuring step consists of moving from one point on the coast to another point lying at a distance of one unit. The number of such steps multiplied by the measuring unit is an approximate measure of the length of the coast. However, coastlines are often so irregular that this measure is a poor approximation. Reducing the measuring unit increases the precision but the number of measuring steps increases so rapidly that the estimated length increases without bound.

Readers interested in the coastline measurement problem should consult Ref. [2]. We merely note here the analogy between the *process* of measuring working sets with time windows of arbitrary size and the *process* of measuring coastlines with rulers of arbitrary length. In both cases, the end result is strongly dependent on the size of the chosen measuring unit.

Furthermore, this measuring process uses a concept of "distance" identical to the topological notion of distance. The mathematical definition of the fractal dimension of a set involves the Hausdorf-Besicovitch measure (or dimension), which is a function of distance in a metric space. This introduces the notion of dimension into the analysis of software working sets as reflected in a conventional cache memory. Our conjecture is that the fractal dimension of a given distribution of cache misses, as developed below, is a measure of the complexity of the underlying software.

Statistical background

The statistical distributions of interest in fractal geometry are hyperbolic distributions of the form

$$P(U > u) = \left(\frac{u}{u_0}\right)^{-\theta}, \tag{1}$$

where very often (depending on the random set under consideration) the exponent can be shown to be identical to a dimension.

To establish the fractal structure of bursts of misses, we must show that the intermiss distance U, expressed as a number of memory references, satisfies Eq. (1) asymptotically (i.e., for large values of u). One way to verify such a behavior is to plot log $P(U > u)$ as function of log u, and to look for a reasonably straight line of slope θ over two or three decades.

The hyperbolic distributions are closely associated with an important concept called statistical self-similarity. Let S be a set of points $x = (x_1, x_2, \cdots, x_n)$. The similarity transformation in geometry transforms point x into point $r(x) = (rx_1, rx_2, \cdots, rx_n)$. Hence the set S is transformed into the set $r(S)$. In the same way, following Ref. [2], a bounded random set S is statistically self-similar, with respect to the ratio r and an integer N, when S is the union of N non-overlapping subsets, each of which is of the form $r(S)$ and has the same distribution as S, except for displacement and/or rotation.

In short: statistical self-similarity involves sets with invariant probability distributions under scaling, and hyperbolic distributions are shown, in Ref. [2], to satisfy the requirements of self-similarity. We model our application of self-similarity and hyperbolic distribution to the analysis of cache reference patterns on the techniques discussed in Ch. 32 of Ref. [2], being motivated by the following conjecture:

If the distribution of intermiss distances is hyperbolic, the bursts are statistically self-similar. That is, any given burst is itself made up of smaller bursts, while bursts themselves come in clusters (bursts of bursts). We identify this structure with the natural layering of software itself; the burst created by a major component is made up of bursts created by individual modules, themselves made up of bursts created by subroutine calls within this module, etc.

In other words, the hierarchical structure of contemporary software projects itself onto a hierarchy of imbedded bursts of misses. Indeed, intuition tells us that this effect will be enhanced by the increased use of "Structured Programming" techniques in software development.

This paper describes initial attempts to use the slope (or dimension) θ to discriminate between different types of workloads by measuring the complexity of the bursts they trigger in the cache. If our interpretation of θ is correct, it may provide a much-sought-after answer to the question of the *intrinsic* difference (if any) between workloads that leads to different cache miss rates. We propose a geometric representation of the bursts of misses, to which the notion of dimension can be associated. Our hope is that the fractal dimension of this representation of cache misses captures an essential relation between program structure and memory organization, thus simplifying the study of software-hardware interaction (see Appendix).

Experiments

We based our initial analysis on three frequently used traces, each of which is representative of a corresponding environment (data-base-interactive, time-sharing, and scientific). These traces are used to drive a simulation model of a 64K-byte cache organized into 128 congruence classes, each containing four 128-byte lines. The machine simulator produced the distribution, over "time" measured in storage references, of cache hits and misses for each input trace under the above cache organization. Figures 1(a)–(c) show the cumulative probability distributions resulting from these simulations (i.e., the probabilities that the intermiss distance U will be bigger than a given value u for the three respective environments). All three graphs are presented in log-log coordinates, and it is clear by inspection that they approximate straight lines over a large range of intermiss distances (indeed, this range is much larger than the span of intermiss distances which we can hope to affect by changes to either hardware or machine architecture).

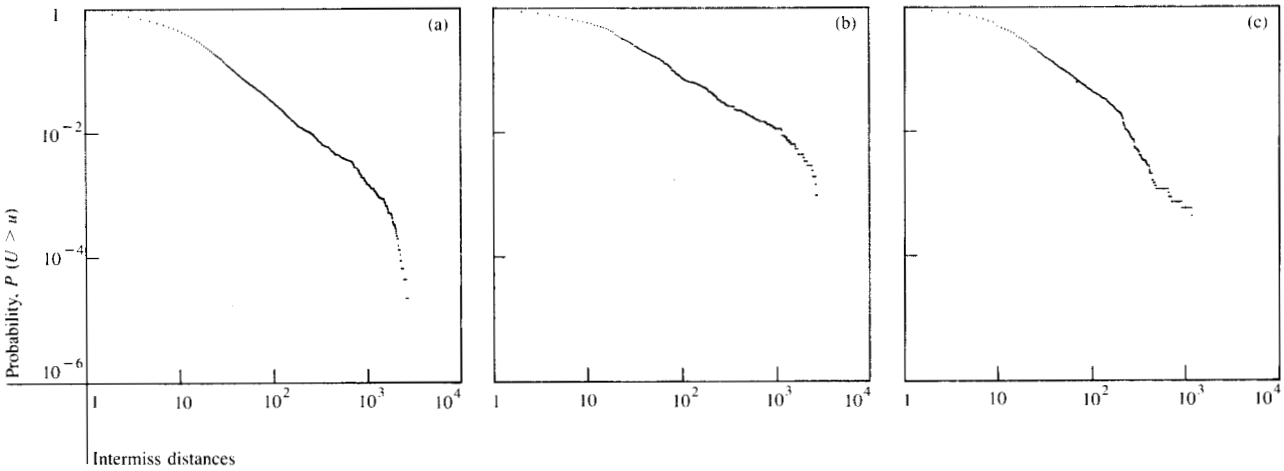

Figure 1 Probability data for (a) data-base-interactive, (b) time-sharing, and (c) scientific environments.

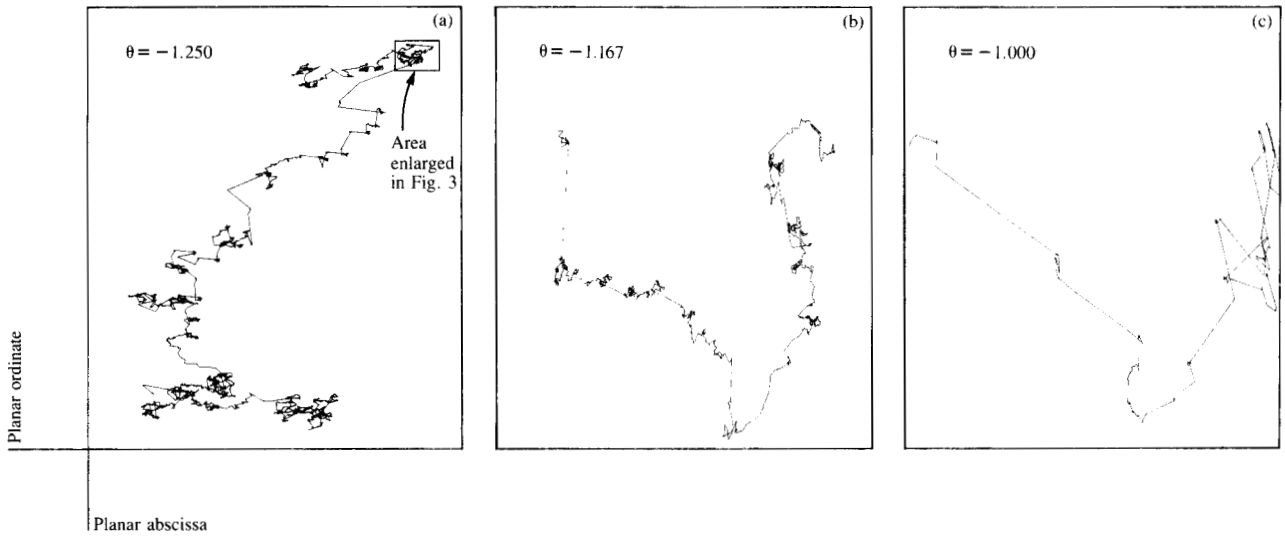

Figure 2 Flights for (a) data-base-interactive ($\theta = -1.250$), (b) time-sharing ($\theta = -1.167$), and (c) scientific ($\theta = -1.000$) environments.

As shown by Mandelbrot in [2], the absolute slope measures the degree of clustering, and is the fractal analog of a dimension. When θ gets smaller, the long intermiss distances get longer and the short intermiss distances get shorter. Thus, the most extreme clustering occurs in the scientific trace [Fig. 1(c)], with very short bursts and very long gaps. Here, the slope has a value of one (computed using a least square fit). There is a degree of arbitrariness to this analysis, since the endpoints of the range over which the curve is deemed flat were estimated—as opposed to algorithmically determined by minimizing the residual error of the least square fit. However, the practical range of interest is constrained more by architectural considerations than by mathematics (possible analysis and interpretation of these boundary values is not pursued in this study).

The data-base-interactive trace [Fig. 1(a)] has a slope of 1.25, which we interpret as indicating a more complex burst structure than the scientific trace. This conforms to our intuitive expectation that data base software generates a far more *scattered* locality of reference than scientific software. Figure 1(b) (time-sharing environment) has an intermediate

slope of 1.167, indicating a somewhat more complex clustering structure than the scientific software, but less complex clustering than the data-base-interactive software.

Random flights

In order to provide a geometric representation of these varying degrees of clustering, we borrow from Mandelbrot the notion of *flights* and *stopovers*, which he uses to describe the "lumpiness" of galactic matter. A flight is a sequence of long jumps, each terminated by a stopover. A "flight" here will be what we have previously called a gap: i.e., an interval of many memory references without any miss. When we encounter a burst, we "stopover" until the reference locality has been absorbed by the cache.

We use polar coordinates to represent gaps and bursts in a plane. A gap (flight or jump) is a line segment represented by the couple (ρ, ϕ). ρ is simply the intermiss distance, in references, for the gap. The definition of ϕ is less straightforward. The congruence class of a line is the least residue, including 0, of the line address and the number of congruence classes (i.e., mod (*line* − *address*, 128) in our machine simulation). We start with an arbitrary angle ϕ for the first jump. Thereafter, ϕ is computed as

$$\textit{congruence class} \times (\pi/64). \tag{2}$$

In his book, Mandelbrot uses a uniform random number for ϕ, whereas we treat ϕ as a measure of locality within the cache itself. If adjacent misses are to sequential line-addresses, their computed directions will be close, so that flights resulting from misses to sequential appear "straight"—and are barely distinguishable from a single, longer flight. Adjacent misses to non-sequential line-addresses result in visibly broken flights, indicating a dispersed locality of reference.

Figures 2(a)–(c) show the resulting graphs for the data-base-interactive, time-sharing, and scientific environments. These pictures are graphic representations of changes of locality. The bursts of misses do appear as clusters, and what we called "gaps" are simple, reasonably unbroken long lines. The result is that the visual complexity of the locality graph for data-base-interactive software contrasts with the visual simplicity of the locality graph for scientific software. Using this graphical presentation, it is easy to "see" the difference between various environments.

The statistical self-similarity of the underlying stochastic process is illustrated by expanding a small part of Fig. 2(a), shown in Fig. 3. Figure 3 represents a part, which looks *similar*, in terms of its overall complexity, to the whole made up by Fig. 2(a).

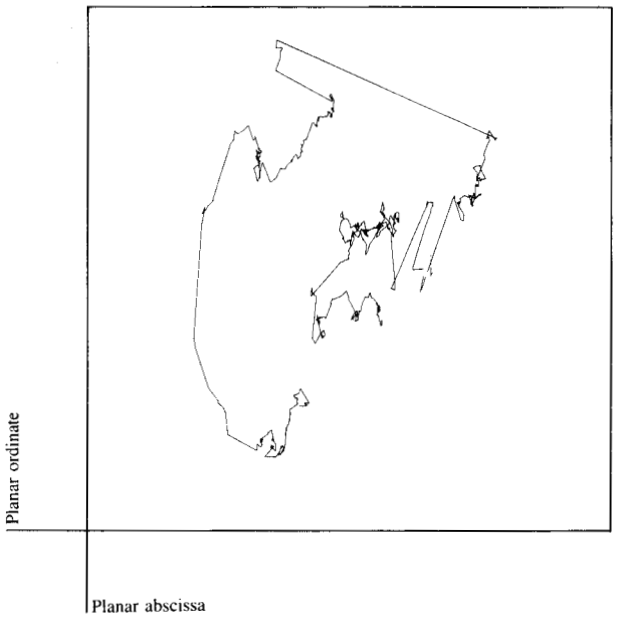

Figure 3 Enlargement of a section of Fig. 2(a) showing the similar random features.

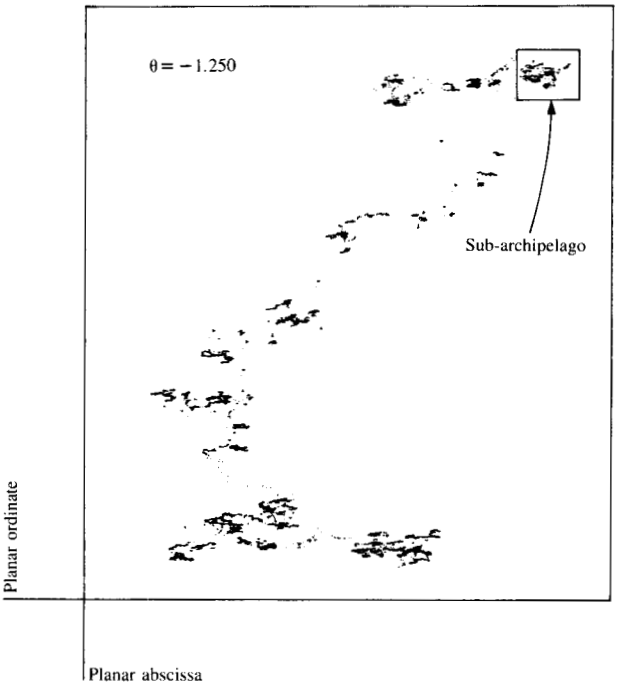

Figure 4 Data-base-interactive archipelagos; $\theta = -1.250$

The data-base-interactive archipelago

Figure 4, which is simply Fig. 2(a) with the connecting lines between points removed, is shown to highlight the structure

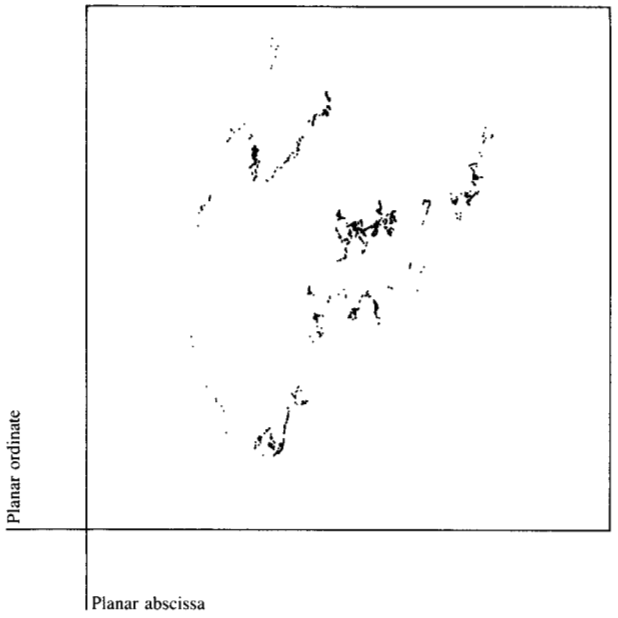

Figure 5 Enlargement of a section of Fig. 4.

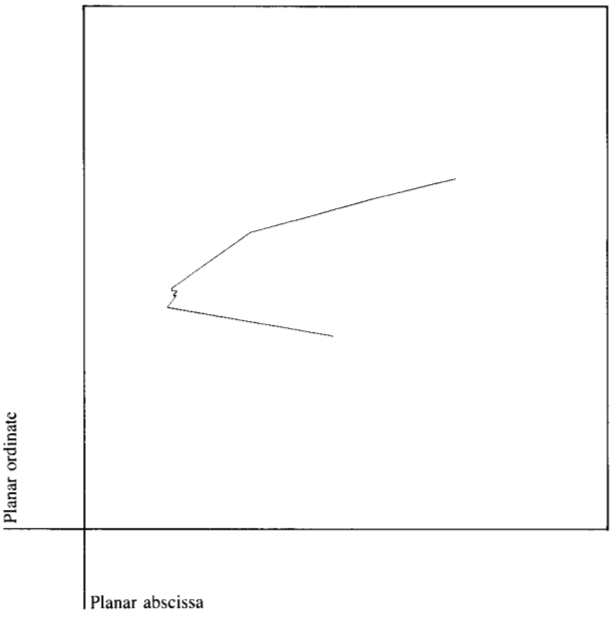

Figure 6 Enlargement of the gaps within an island.

of the miss clusters, rather than that of the gaps. The result looks very much like a set of clustered islands, or an archipelago.

The self-similarity property of these islands is shown in Fig. 5, which is an enlargement of one cluster (arbitrarily selected). Both exhibit similar random features.

More interesting, however, is the analysis of the software events associated with each "island" of cache misses. We were able to identify the bursts forming this island with invocations of various modules listed in the memory map for the benchmark. The island begins to form at memory reference number 110 071 and ends at reference number 182 049—i.e., it is 71 978 references "long" on a straight time-line. During its life, the island exhibits a buffer miss ratio of 4.72%, in comparison to the overall buffer miss ratio of 4.19%. Moreover, the island is followed by a flight (gaps) beginning at memory reference number 182 050 and ending at reference number 199 894. This flight has, as expected, a much lower cache miss ratio of 0.43%. It consists of three long gaps (respectively 4224, 3976, and 5339 memory references), and is shown enlarged in Fig. 6. The tracing program indicates that the flight corresponds (for the most part) to user code whose locality of reference is easily captured by the cache.

By contrast, the "sub-archipelago" islands (Fig. 5) have a much more complex locality. A detailed module-by-module analysis reveals the conventional transfer pattern: 1) entry into the input/output supervisor; 2) entry into the buffer handler of the data base subsystem; 3) calls to a search function; and 4) showers of misses triggered by the return to the user code.

Obviously, this sub-archipelago corresponds to active transfers between the data base subsystem code and the operating system code. This underscores the fact that while getting a segment from a data base into the user's area is a simple concept, it is a very complicated process. This is one reason why data-base-interactive loads have intrinsically higher buffer miss ratios than most other types of work. Seen from a different viewpoint, the user interface in this data base system is a high-level interface, the simplicity of which hides the complexity of the task to be performed on the system side of the interface.

Within the 71 978 memory references in the sub-archipelago, there are 2512 module invocations, which amounts to 28.65 references per invocation. Given the inherent cost (in references) of linkage between modules, only a very small quantity of useful work done remains per module invocation! Even if this overhead could be reduced by repartitioning (an open question to which there are few ready answers), it is clear that a complex sequence of relatively simple functions is needed to complete execution of typical data-base-interactive tasks.

Another definition of the working set

We define a working set as being the number of distinct cache lines referenced in a gap. A more exact definition would be to include in this window the burst preceding this gap. It is, however, straightforward to compute working sets in any intermiss distance, and only to consider the subset of these numbers which correspond to gaps that are big enough to belong to the "straight" (i.e., fractal) part of the probability distribution.

Computations were made for all three traces; these were checked to see if the working set sizes correlated, in any way, with the corresponding gap sizes. They indeed do. The correlation coefficients for the interactive, time-sharing, and scientific environments are 0.773, 0.794, and 0.906, respectively. Let WS be the working set size expressed in a number of cache lines; the corresponding relationships for the three traces are

$$U = A(WS)^B, \tag{3}$$

where the respective values of A and B are: interactive—1.09, 1.50; time-sharing—1.59, 1.34; scientific—1.30, 1.49.

Therefore, if the intermiss distances are fractally distributed (which implies a power law for their probability distribution), so are the corresponding working set sizes. This makes the working sets statistically self-similar, and subject to a power or hyperbolic law. We can now use this result in the following. Figures 1(a)–(c) show that the linear relationship breaks down at some threshold value. For the database-interactive trace, this value is around 3000 references. The question is raised as to the meaning of this physical discontinuity. From the relationship between intermiss distance and working set size, a first answer links the threshold value to half the size of the simulated cache; $1.09 \times (3000)^{2/3} = 225$ lines, where the size of the cache is 512 lines.

This seems to imply that working sets are unlikely to be larger than half the size of the cache, for this type of workload and the specific cache simulated in our experiments.

Questions

This method of differentiating between workloads raises many questions. For example, a batch trace analyzed in the same way did *not* produce an intermiss distribution with an asymptotic straight line. However, when another simulation was run using a much smaller cache of 16K bytes, the distribution did exhibit an asymptotic straight line, with a slope equal to 1.4. This suggests (but is not sufficient to prove) that when the cache is not stressed, the distribution of intermiss distances is no longer hyperbolic—exhibiting a more regular behavior corresponding to a thin-tail versus fat-tail distribution.

Further experimentation is required to determine the influence of the cache design parameters on the existence of a "straight line" hyperbolic miss distribution and the value of its slope. In particular, the (non-hyperbolic) distributions of intermiss distances for very large caches may not follow the model presented here.

Conclusions

In summary, we propose a geometry of software behavior, which visualizes changes of locality, and helps to pinpoint problems in software taxonomy.

The revised notion of a working set is helpful in thinking about memory hierarchy design. Our notion of working set is defined by the memory hierarchy studied and by the software involved. It is therefore more intrinsic to the problem, having no dependence on an *a priori* window. A similar concept of "resident" set is defined for paging environment by Bard in Ref. [4].

This use (and possibly abuse) of the notion of dimension appears to be an effective measure of software complexity, where complexity is understood as *structural* in contrast to the classical *computational complexity*. The self-similarity property intuitively evokes a natural hierarchy of working sets, each induced from a previous level by a self-similarity ratio defined by the workload characteristic. It is significant that the model itself "involves no explicit hierarchy, only built-in stochastic self-similarity" [2].

Finally, we are satisfied that the interaction of software with memory hierarchies is naturally modeled by a stochastic process having a larger than usual amount of irregularity. The degree to which this phenomenon is captured by our fractal model in turn implies that it belongs to a very large class of natural processes.

Appendix

Reference [2] goes in depth into a discussion on what the proper definition of a fractal should be. As we stated earlier, this paper is neither an introduction nor a treatise on fractals.

However, for interested readers we excerpt from Ref. [2, p. 361], that a fractal set is a set, in a metric space, for which the Hausdorf-Besicovitch dimension is bigger than its topological dimension.

The exponent θ has been shown, for many specific sets described by hyperbolic distributions, to coincide with the Hausdorf-Besicovitch measure (dimension) of the set.

In this paper, we abuse the language (and, perhaps, the state of the art), by calling all hyperbolic exponents "dimensions." This terminology both fits our intuition and helps us make the geometric interpretation specific.

References

1. J. Voldman and L. W. Hoevel, "The Software-Cache Connection," *IBM J. Res. Develop.* **25,** 877–893 (1981).
2. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman and Co., San Francisco, 1982.
3. P. J. Denning, "The Working Set Model for Program Behavior," *Commun. ACM* **11,** 323 (1968).
4. Y. Bard, "Characterization of Program Paging in a Time-Sharing Environment," *IBM J. Res. Develop.* **17,** 387–393 (1973).

Received September 22, 1982

Lee W. Hoevel *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Hoevel joined IBM in 1978 as a Research staff member at the Thomas J. Watson Research Center. He is currently a member of the experimental systems structure group and is involved in the analysis of cache performance and the design of system extension mechanisms. Dr. Hoevel completed his undergraduate work at Rice University, Houston, Texas, in 1968, and later received a Ph.D. in electrical engineering from the Johns Hopkins University, Baltimore, Maryland, while a Research Associate at Stanford University. Dr. Hoevel is a member of the Association for Computing Machinery, the Institute of Electrical and Electronics Engineers, and Sigma Xi.

Joshua Knight *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Knight joined IBM in 1981 as a Research staff member. From 1978 to 1981 he was a research associate at Stanford University, California. Dr. Knight received a B.S. in engineering physics from Cornell University, Ithaca, New York, in 1968 and a Ph.D. in applied physics from Stanford University in 1978.

Benoit Mandelbrot *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Mandelbrot, an IBM Fellow at the Thomas J. Watson Research Center, is best known as "the father of fractals" and the author of the books, *Fractals; Form, Chance and Dimension,* 1977, and *The Fractal Geometry of Nature,* 1982. He is a graduate of Ecole Polytechnique in 1947, California Institute of Technology, Pasadena, with an M.S., in 1948 and an Ae.E. in 1949 in aeronautics; and the University of Paris, with a Docteur es Sciences Mathematiques, in 1952. Dr. Mandelbrot's first positions were with the French National Center of Scientific Research, the Institute for Advanced Study, under J. von Neumann, and the University of Geneva. Before joining IBM in 1958, he was a junior Professor of Applied Mathematics at the University of Lille and of Mathematical Analysis at Ecole Polytechnique. On leave from IBM, he has been a Visiting Professor of Economics, later of Applied Mathematics, and then of Mathematics at Harvard University (1962 to 1964 and 1979 to 1980), of Engineering at Yale University (1970), and of Physiology at the Albert Einstein College of Medicine (1971). He has visited the Massachusetts Institute of Technology in 1953, 1956, 1964 to 1968, and most recently as an Institute Lecturer, the Institut des Hautes Etudes Scientifiques (Bures), and the College of France. Before turning to fractals, he worked on the theory and applications of stochastic processes, thermal and other fluctuations in electrical systems, thermodynamics and turbulence, problems of communication, particularly through natural languages, economic time series (especially with regard to commodity and security prices), and the general theory of self-similar or sporadic chance phenomena. Dr. Mandelbrot was a National Lecturer of Sigma Xi and the Scientific Research Society of America. He was a Fellow of the Guggenheim Foundation, and is a Fellow of the American Academy of Arts and Sciences, the Institute of Mathematical Statistics, the American Statistical Association, the International Statistical Institute, the Econometric Society, and the Institute of Electrical and Electronics Engineers. He is also a member of the American and French Mathematical Societies and the American Geophysical Union.

Philip L. Rosenfeld *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Rosenfeld, who joined IBM in 1978, is currently manager of multiprocessing cache studies. His interests include operating systems, multiprocessors, and memory hierarchy behavior. Dr. Rosenfeld received his B.S. in engineering from Cornell University, Ithaca, New York, in 1973 and the M.S. in 1975 and the Ph.D. in 1980, both in operations research from Cornell University.

Jean Voldman *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Voldman is currently senior manager of experimental system structures within the systems laboratory in the Computer Sciences Department at the Thomas J. Watson Research Center. He is interested in defining hardware and software structures to support large transactions processing systems. Mr. Voldman joined IBM France in 1968, working as a system programmer for IBM France information systems. In 1970, he was named manager of the systems programming area in Orleans, France. In 1973, he joined the Advanced Systems Development Division in Mohansic, New York, as an international assignee, working as a product planning advisor. In 1976, he came back to information systems in IBM France at the Orly location, where he worked as manager of systems programming. He then left in 1978 for his present international assignment in the United States at Yorktown Heights. Mr. Voldman graduated from Ecole Nationale Supérieure de Mécanique et d'Electricité (ENSEM) in Nancy, France, in 1965. He holds a License es Sciences Physiques from the University of Nancy.