THE FRACTAL STRUCTURE OF DATA REFERENCE

Applications to the Memory Hierarchy

THE FRACTAL STRUCTURE OF DATA REFERENCE

Applications to the Memory Hierarchy

by

Bruce McNutt
International Business Machines Corporation
San Jose, California, U.S.A.

KLUWER ACADEMIC PUBLISHERS
New York / Boston / Dordrecht / London /Moscow

eBook ISBN: 0-306-47034-9
Print ISBN: 0-792-37945-4

This book is dedicated to the system programmers who, as survey participants, donated their time and skills to advance everyone's understanding of disk workloads.

Contents

List of Figures

List of Tables

Preface

For purposes ofunderstanding its performance, a computer system is traditionally viewed as a processor coupled to one or more disk storage devices, and driven by externally generated requests (typically called *transactions).* Over the past several decades, very powerful techniques have become available to the performance analyst attempting to understand, at a high level, the operational behavior of such systems.

Nevertheless, the ability of computers to rapidly deliver requested information, at the time it is needed, depends critically on an underlying structure whose operational behavior is not nearly as well understood: the *memory hierarchy* through which data stored on disk is made available to the processor. The memory hierarchy tends to consist of many levels, including one or more processor buffer memories (also called L1 and L2 cache), main processor memory, disk storage cache memory? disk storage, and often tape. A given level is usually configured to have both a much smaller storage capacity and a much faster access time than the level below it. The data stored in each memory level is managed dynamically, with new data being brought in as needed from the levelbelow.

The architectural concept of a memory hierarchy has been immensely successful, making possible today's spectacular pace of technology evolution in both the volume of data *and* the average speed of data access. As we shall find in Chapter 1, however, its success is difficult to understand if we rely upon a traditional performance modeling framework. This is because the power of a memory hierarchy comes from its ability to take advantage of patterns of data use that are *transient.* Repeated requests tend to be made to a given data item over a short period of time, after which the data may remain unreferenced for extended periods (or even indefinitely). Such patterns do not yield easily to an analysis based upon steady-state, "memoryless" behavior.

This book adopts a *heavy-tailed* distribution for the time between requests to a given data item: substantial probabilities are assigned even to very long interarrival times. The probabilities are sufficiently high, and the corresponding gaps between arrivals are sufficiently long, that the average time until the next reference becomes unbounded. There is *no* steady-state arrival rate. This modeling approach reflects the opposite of the usual steady-state assumptions: arrivals are attributed to transient events, rather than to an ongoing arrival process.

In a reflection of its emphasis on transient events, the proposed modeling approach calls for a very high arrival probability during the time shortly after a request. The probability then falls off rapidly, if no request occurs. This results in a pattern of accesses characterized by short "bursts", separated by long periods of inactivity. By studying such transient "bursts", we are able to gain a practical understanding of memory hierarchy performance.

Heavy-tailed distributions are characteristic of fractal, or self-similar, phenomena. As discussed in Chapter 1, the presence of self-similarity in data reference patterns should not be surprising; indeed, it would be reasonable to anticipate it based upon the explicitly hierarchical nature of most software. To reflect the inferred "behind the scenes" role of hierarchically structured software, we adopt, in Chapter 1, the term *hierarchical reuse model* to describe the overall modeling framework which this book proposes. We also show that the proposed heavy-tailed distribution provides an excellent approximation of realistic interarrival times to individual data items.

Even without the arguments of Chapter 1, self-similar behavior would *still* be plausible, or even likely. Such behavior has now been observed in *many* aspects of data reference [1, 2, 3, 4, 5, 6, 7]. The pioneering work by Voldman, Mandelbrot, and others [1], and later by Thiébaut [2], is particularly worth noting, since it established that the sequence of locations referenced by software tends to be structured in a self-similar manner. This fact certainly enhances the plausibility of a self-similar relationship involving interarrival times to any single, identified reference location.

Despite the importance of self-similar phenomena in describing the operational behavior of computer systems, however, this preface is perhaps the place to admit that the demonstration of one more example of such a phenomenon is not, by itself, sufficiently interesting to justify an entire book. Instead, the "meat" of the present book comes from the ability of the hierarchical reuse model to impose a mathematically tractable structure on important problems involving the operation and performance of a memory hierarchy. We shall obtain striking insights into how such hierarchies work, and how to exploit them to best advantage. The emphasis of the book will be on the practical applications of such results.

The focus of the applications examined in this book is on the portion of a computer system's memory hierarchy that lies below the level of the "get" and "put" statements issued by running software. This includes file buffer areas in main processor memory, cache memory associated with the disk system, disk storage, and tape when it is used as a repository for dynamic migration and recall.

The primary audience intended for the book is the large body of workers who endeavor to ensure that computer systems, and the associated disk storage, are configured and managed so as to deliver the expected and desired level of performance. This includes practitioners in the area of computer performance management and capacity planning, as well as those who build tools to facilitate their work.

To apply many of the ideas presented in this book, performance practitioners will require some minimal level of storage subsystem performance reporting (for example, measurements of cache hit ratios). On subsystems running under the Operating System/390 (OS/390) and Virtual Machine (VM) operating systems, the needed reporting is universal and standardized. For this reason, these environments are the main source of the case studies, empirical data, and examples presented in the book. Performance practitioners working on the Unix and Windows NT platforms will also find, in many cases, that the needed performance reporting can be obtained. Although it is not standardized across storage vendors, vendors often offer such reporting as a differentiating feature.

As befits a work about fractal modeling, the chapters of the book cover events that play out at a wide range of time scales. The first five chapters focus on the operation of file buffers and disk cache; memory residency times at this time scale range from a few seconds to several minutes. Among these five chapters, those seeking a quick introduction to the most important ideas should select Chapter 1 (fundamental modeling concepts), Chapter 3 (capacity planning for cache memory), and perhaps Chapter 2 (a concrete illustration of hierarchical reuse behavior). Chapters 4 (interaction of processor file buffers and storage control cache) and 5 (cache memory management) cover additional topics important to performance specialists.

We then gradually lengthen the time scale. Chapter 6 considers the collection of free space in a log-structured disk array; here the time scales range from minutes to days. Chapters 7 and 8 examine the life cycle of disk storage files, and their management via migration and recall; this involves time scales ranging from hours to months. Chapter 7 also explores the limits of the hierarchical reuse model, by identifying patterns of data item use which appear to be more persistent than the hierarchical reuse model would suggest.

Finally, Chapter 9 develops a fresh way to look at the planning for disk storage, based upon viewing entire *applications* from a statistical standpoint (as though they were objects belonging at some deep level within a memory hierarchy, operating on a time scale of months to years). This allows important insights to be gained about how to strike an overall balance between disk storage capacity, cost, and performance.

BRUCE MCNUTT

Acknowledgments

The author would like to acknowledge the vital roles played by Mr. Jeffrey Berger, Mrs. Neena A. Cherian, Mr. Joe Hyde, and Mr. Charlie Werner, each of whom developed key tracing or trace analysis software without which the investigations reported in the present book would not have been possible.

In many stimulating discussions, at the annual conference of the Computer Measurement Group and elsewhere, individuals have shared their views and comments on the topics presented in the book. This input was invaluable in organizing the presentation of ideas. The author would like to thank all of those who did so, including especially Dr. H. Pat Artis, Dr. Alexandre Brandwajn, Mr. Mark Friedman, Dr. Gilbert Houtekamer, Mr. David L. Peterson, and Mr. Brian J. Smith. Thanks also to Mr. David A. Short and again to Brian Smith for their help with editing the manuscript.