Technology Projection Modeling of Future Computer Systems

Al Cutaia

Prentice-Hall, Englewood Cliffs, NJ, 1990
280 pages
ISBN 0-13-898479-4

Perhaps nowhere has the pace of technological change been more evident than in the computer industry. As such, those involved in forecasting, planning and developing computer systems, as well as those looking to take advantage of capabilities provided by future computer systems, urgently need a way to influence and exploit those changes. This book provides such a way, and it does so in a concise, yet thorough, manner.

The text is well-written and well-illustrated, and progresses in an on organized, logical sequence. Chapters 1 and 2 of the book discuss the power of technological changes and their influence on computer designers, manufacturers, technologists, and end users. Chapters 3 and 4 present a selection of key technological drivers that will affect the development of mid-range and entry systems through the year 2000. Chapters 5 through 7 develop a system model and show the generic effect of technology changes on that system model. Chapters 8 through 10 describe the projected effects of new software applications enablers on current computer system hardware and software. The chapters include a discussion of parallel systems architectural designs, showing the significant possible mid-range system performance possibilities. Chapter 11 ties all the preceding material together, summarizing modeling results for the years 1987 to 2000 from a total systems view point. Appendix A presents hardware specifications used to develop the generic system model in chapter 6. These specifications change over three time periods: 1987-1991, 1987-1995, and 1987-2000. Appendix B presents the approximation model used to make the qualitative projections about how new technologies will affect computer system response times. Appendix C presents the model used to analyze the performance and capacity of the projected system model. The book closes with a good reference list, and a thorough index.

The book is written for those interested in future views of system architecture, system design, and system component technologies. As intended, it will likely be very useful to (1) students in computer science or engineering focusing on computer architecture, (2) professional engineers interested in advanced hardware and software design, (3) computer planners and strategists, (4) engineers interested in component technology developments, and (5) computer users interested in future systems architecture and new applications. The book succeeds in accomplishing its two major objectives. First, it provides a methodology for computer systems development forecasting. This methodology includes performance evaluation tools that enable quantitative evaluation of how technological changes will affect system attributes. Second, the book provides a flexible method for maintaining a continually updatable understanding of expected technological advances through the year 2000. In the book, the author focuses on entry and mid-range systems, although the techniques he presents can be applied to the analysis of other computer systems.

The approach taken in Cutaia's methodology differs from those of previous authors. Cutaia directly relates detailed technology changes to changes in system architecture and structure. He avoids the general notions that future technology changes will merely be linear extrapolations of the past and that future system structures will be similar to those we hold dear today. Instead, he goes a step further, and assumes that future technologies will affect both the developer's and end user's views of future computer systems. In this way, system planners can accurately forecast not only what will be technologically feasible but also what will become marketable. For example, the technology calendar described by the author in chapter 4, is a table that shows the expected changes in component technologies during a selected time period. Such a tool can be used to determine the relative position of competitors and which key technology drivers are critical for an existing or future computer system.

This book is a well-written and useful work. And it comes at a time when it is much-needed. Those concerned with the pace, direction and impact of technological changes on the computer industry ought to have a copy.

reviewed by:
Keith Anthony
2278 Maryland Drive
Xenia, OH 45385-4663

Optimizing FORTRAN Programs

C.F. Schofield
Halstead Press, 1989

As a practitioner of compiler optimization (ie., one who has written optimizing compilers), I was disappointed when I discovered that "Optimizing Fortran Programs" is written for users of high-speed computers and not for authors of compilers. The title may result in the book not being used by the audience for which it has the most utility. That would be unfortunate, for this 300-page volume can be most useful for large number of users eager to get maximum performance from their computer systems.

Although the date of copyright of this book is 1989, it appears to have been written no later than 1984 (with occasional anachronisms). Consequently, subsequent advances in compilers have not been reported. This makes some of the advice unnecessary and a little bit of the advice counterproductive.

The author uses five machines as representative of different classes of computer systems/computation: The CRAY 1, the CYBER 205, the Amdahl 470 (which also represents the IBM 3770), the ICL 3980, and the Apple Macintosh. Most of the source program writing techniques are described in terms of their effect upon each of the target machines and in terms of execution time in a "before and after" comparison.

Chapter 1, the Introduction, sets the background and describes the context in which code optimization is done. Although Amdahl's Law is usually taken to refer to parallelism, a variant is illustrated to show the affect of optimization on different fractions of code.

Nom Chapter 2, General Optimization, describes well known machine-independent optimization techniques. These include constant propagation, common sub-expression evaluation, code motion, and strength reduction,. This chapter should be useful to all but the most knowledgeable scientific programmers.

Chapter 3, Storage, explains the concept of banked memory and its effect on storage access time. Programmers are told how to avoid access patterns which cause bank conflicts. Paging and caching are similarly described.

Chapter 4, Loops, begins with the generally inaccurate statement that "Complex loop structures involving if and go to statements should be avoided (as they are) ...unlikely to be recognized as a loop by most compilers, and will therefore not be optimized". This section of the book, although useful, is rather dated. References suggesting compilation in FORTRAN 66 are not useful today. The discussions of loop interchange and contiguity are valuable. Although "parallel computers" (Hockney and Jesshope) is among the references, it appears not to have contributed to the author's explanation of the effects of vector length on speed. A lengthy analysis of loop unrolling follows. For the careful patient reader, many pointers are offered. The chapter includes with a discussion of branching within loops. Unfortunately, recent compiler advances, such as those found in the Fujitsu compiler, with its use of the "true ratio", readers are not mentioned.

Chapter 5, Vectorization of Loops, discusses material in the context of the CRAY 1 and the CYBER 205. In 1989, these machines are already obsolete. These general principles ok vectorization remain unchanged. However, the specific timings are of little value. Additionally, the space given