Reminiscences on Influential Papers

Kenneth A. Ross, editor

I continue to invite unsolicited contributions. See `http://www.acm.org/sigmod/record/author.html` for submission guidelines.

Mary Fernandez, AT&T Labs-Research, `mff@research.att.com`.

[David Patterson and David Ditzel. The Case for the Reduced Instruction Set Computer. Computer Architecture News, Vol 8, pages 25–33, 1980.]

As the title implies, the article is a position paper describing the observed and anticipated benefits of RISC architectures. In the CISC past, memory was very expensive and woefully slower than CPUs. This imbalance demanded that programs be compact and complex subroutines be implemented in microprogrammed control. Implementing complex software systems in assembly language was still common practice and "instruction selection" was performed by the bleary-eyed programmer. RISC bucked traditional architecture design by taking advantage of faster memory and new VLSI technologies and by supporting the development of compilers for high-level languages that could select instructions and allocate registers (close to) optimally.

The article initially impressed me because it described the exasperating tasks that I performed working as an undergraduate "SPOC" (student programmer, operator, and consultant). These chores included loading microcode patches from floppies only to find that they didn't fix a particular bug, and pouring over manuals trying to find the right string-manipulation operator while writing assembly code for, among others, the IBM 360, PDP-11, and VAX 11/780. (Given those morsels of information, this writer's age is left as an exercise to the reader.)

Reading between the lines, the article reveals deeper principles of system design. RISC demonstrated that a minimal and orthogonal set of core instructions, chosen to support the common behavior of most programs, yields efficient, flexible, and versatile architectures. The critical task of choosing the best set of instructions for a program is left to a program, not a person. These principles can be generalized to almost any system and are especially germane to language design. Well-designed query (and programming) languages contain a core set of orthogonal operators that can be combined to support higher-level user constructs, and query optimizers (or compilers) search for the best execution plan. In the design of any system, however, the first, and possibly hardest, problem is identifying that minimal and orthogonal core.

Kyuseok Shim, Seoul National University, `shim@ee.snu.ac.kr`.

[Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, Thomas G. Price. Access Path Selection in a Relational Database Management System. SIGMOD Conference 1979, pages 23–34]

I first met this paper when I began to look for Ph.D thesis topic. As I read the paper, I was amazed at how the dynamic programming approach could be applied for the query optimization problem elegantly. Furthermore, it taught me several essential techniques to build query optimizers including the estimation of cardinalities of the intermediate relations, interesting orders and processing of subqueries. I actually switched to Computer Science from Electrical Engineering for graduate study after I was fascinated with the beauty of designing algorithms with the maximum amount of efficiency. Thus, I was convinced that query optimization is the right topic for me to pursue. Immediately, I rushed in and fell in love with the query optimization problem. In retrospect, I think I was very lucky to run into the paper.

While working as a summer intern in HP Laboratories, I was asked to implement a simple prototype of System-R style optimizer by my mentor, Surajit Chaudhuri. It gave me the great opportunity to understand the details of the System-R optimizer in depth and it helped my future researches on the problems of optimizing SQL queries in the presence of group-by, user-defined predicates, materialized views and nested subqueries. In each of these problems, the basic and essential elements of the System-R optimizer were nicely generalized in our proposed algorithms over and over again. The paper also influenced significantly my subsequent work including data mining algorithms. Thanks to the experiences with the paper, I cover the details of System-R optimizer in my undergraduate database lectures. It always reminds me of the exciting moment that led me to the field of query optimization.