Turning points in systems architecture

by A. G. Ganek
E. H. Sussenguth

In order to discuss papers representing turning points in systems architecture that have appeared in the *IBM Systems Journal*, our first challenge was to come up with a workable definition for systems architecture. The meaning of "architecture" as used in computer systems is normally credited to Gerritt Blaauw, who in the late 1950s, when at IBM, emphasized the desirability of a sharp distinction between a logical structure and various physical realizations of it.[1] Through the years the interpretation of what architecture means has been expanded to at least three definitions: (a) a direction-setting concept, (b) a set of products following that concept, and (c) a set of rules to which products must conform so that they may interoperate or present a consistent interface to the product user, or do both. Defining "systems," however, has proved far more problematic. The concept of what comprises a system has evolved dramatically over the last four decades. Initially, systems were thought of in a largely central-processing-centric way, consisting of a single collection of a central processing unit (CPU), memory, and I/O devices, whose architecture was largely comprised of the addressing scheme, register population, and CPU instruction set as the key architectural elements.[2] This view has expanded tremendously over the years, and a "system" can now be associated with virtually any collection of computers and related computing resources and enabling software, however geographically dispersed, that are interconnected in some coordinated way so as to perform some useful function. Systems architecture for our purposes relates to the underlying structures that support systems in this broad definition, and that adhere to the three architecture precepts we listed above.

In this light, we believe there are numerous aspects of systems architecture, which, among many others, include: processor architecture, multiple processor complexes, memory structure, I/O subsystem, operating systems, distributed processing services, databases, and networking.

Clearly, a complete treatment of any of these topics could fill volumes, and many have been the focus of entire issues of the *IBM Systems Journal* and of numerous fine individual papers. We will highlight only a small number of representative papers on these topics, with just a few reprinted in this section. The reader should note that these reprinted papers span the years 1973 to 1995 and that they should be read in the context of the time within which they were written.

Of the many systems architecture milestones achieved in the four decades starting with the 1960s, we believe the following seven stand out as chronicled in the *IBM Systems Journal*. The reprinted papers mentioned as we go through these milestones represent some particularly significant turning points.

System/360

As a family of processors, the System/360* implemented a single instruction set across a line of processors of varied performance levels to allow the same program to be run in a variety of environments. With the introduction of System/360 in 1964, IBM was the first major computer company to base an entire line of pro-

cessors on this fundamental principle. The System/360 and its System/370* and System/390* successors have been the cornerstones of the IBM product line ever since then. In a paper published that year, Blaauw and Brooks[3] describe the logical structure of the six processors announced at that time by focusing on the details of data formats and instruction set, as well as the functional structure of the system. Indeed, they state in the opening paragraph: "The overall structure lends itself to programming-compatible embodiments over a wide range of performance levels." What better definition of processor architecture can there be? The entire issue of Volume 3, Numbers 2 and 3, 1964, of which the paper is a part, completes the Blaauw-Brooks discussion, much of which is still incorporated and upwardly compatible to IBM's current System/390 (S/390*) mainframe computers, now implemented with state-of-the-art microprocessors.

Virtual storage

Initially created by the Atlas system built in England in the 1950s,[4] virtual storage masked the difference between the program address space apparent to the programmer via the instruction set and the physical address space available given the memory constraints of the system. This innovation made the task of programming, both for applications and operating systems, substantially simpler. This basic idea utilizes dynamic address translation that enables a hierarchy of memory and storage devices to appear to the program as a uniform addressable memory space. Also, another significant advantage is for multiprocessing environments, where each process may be assigned its own address space, protected from other processes, allowing more addressable memory and improving reliability. Marc Auslander and Joan Jaffe in the paper "Influences of Dynamic Address Translation on Operating System Technology," reprinted here, discuss the concepts of virtual storage as understood and implemented in 1973. The principle is straightforward: make data access simple. The details are not, and the referenced paper provides an excellent tutorial on virtual storage, dynamic address translation, paging, the history, and implementation.

AS/400

Although the AS/400* was the outgrowth of prior IBM systems, such as the System/36* and System/38*, it represents a highly integrated set of characteristics, including a formal architecture constructed in hardware and software layers beginning with a processor hardware core, a high-level machine interface implemented microcode, operating system function, communications support, application enablement including an integrated relational database, and end-user interfaces. The machine interface and layered structure allowed the AS/400 to make a transition from the original complex instruction set computer architecture implementation to the PowerPC* reduced instruction set computer architecture without requiring recompilation of applications, which is an impressive achievement. Data abstraction is achieved via object-based orientation. The concept of virtual storage described above is extended in the AS/400 to achieve single-level storage. Although the System/370 architecture and most other virtual storage architectures enable memory addressability to exceed the physically available memory space, a programmer is still responsible for understanding the distinction between permanent files in external storage devices and temporary data in memory during program execution. Knowing this aspect of virtual storage as they write applications, programmers must include steps to ask the operating system to bring data into memory if not already there, after which their programs may continue. In contrast, the AS/400 is a single-level-store machine: programmers do not know where their data reside, and references to files in the program implicitly cause a combination of hardware and software to make all of a programmer's data appear to be immediately accessible. Files appear to reside in one or more memory segments, and the paging mechanism is used to bring data in and out of memory. This unique system is described in "System Overview of the Application System/400" by David Schleicher and Roger Taylor, reprinted here.

RISC processors

The S/390 and the original AS/400 instruction sets just described represent examples of complex instruction set computers that implement instructions geared to higher-level application functions. A reduced instruction set computer, or RISC, provides a contrary approach based on the observation that complex instructions are rarely used as compared to the simpler instructions, yet they may complicate and, therefore, slow down

the machine or increase the cost of the machine. A RISC processor utilizes only simple instructions and optimizes hardware real estate for performance, allowing low-level parallelism in which more than one instruction can execute per cycle. More complex function is implemented via software, but because it is generated by compilers, it should not increase complexity for the programmer. The design principles and concepts of the RISC were described by Hopkins[5] in 1987. The 801 project launched RISC architecture, now widely available from IBM in its RS/6000* family, as well as from several other manufacturers. Even though RISC technology had its initial successes in scientific and technical computing, it is now in broad use for all kinds of computing environments. Ironically, as indicated earlier, current implementations of the AS/400 now use a RISC core as the hardware processor engine.

Distributed processing and networking

In the 1970s small processors burst onto the computer scene. Whether called small computers, minicomputers, microcomputers, or personal computers, their impact is possibly greater than any other computer innovation. They led initially to distributed processing in banks and retail stores, and now to processing on almost every desktop, in the home, and in numerous mobile applications such as personal data assistants, intelligent cellular phones, and other small devices. The ability to do processing away from the so-called "glass house" of the 1960s and early 1970s required that distributed small (and large) processors be networked. In the early 1970s the application programs written to support distant terminals were, in general, incompatible with one another, and it was impossible to share resources (for example, a telephone line) between two or more applications. Moreover, the distributed processing capability arising at that time would have required those programs to be rewritten, because the programmers who wrote the programs had assumed the remote devices had no computational power (i.e., were "dumb terminals"). In 1974 IBM introduced Systems Network Architecture (SNA) and several associated products to provide a consistent framework for applications, to support distributed processing, and to allow the sharing of communications resources. Although SNA is still in substantial use today, the Internet with its associated standards is the vehicle of choice for most new networking applications. Unfortunately, space limitations do not allow a paper on SNA to be reprinted in this section, but the interested reader is referred to the following subset of networking papers from the *IBM Systems Journal*: an overview of SNA by McFadyen;[6] network architectures by Green;[7] SNA networking by Gray and McNeill;[8] and SNA routing by Jaffe, Moss, and Weingarten.[9]

In the paper "Distributed Data Processing," reprinted here, Allan Scherr provides an analysis of the motivations for and issues involved in distributed processing, including many of the design trade-offs that might be considered. He describes the potential benefits and pitfalls of distributed processing. Together with a similar tutorial paper by Lorin[10] from 1979, the reader may gain significant insight into the rationale of why a corporation may elect this system structure (response time, availability, incremental growth, load balancing) and the difficulties of achieving it (communications costs, error recovery, networks). Scherr's paper, although published long before the Internet explosion of the past few years, presents several models for distributing application function among "front-end" and "back-end" systems that relate closely to today's intelligent "front-end" Web browsers. Scherr went on to refine these ideas in subsequent papers, including one on structures for networks of systems.[11] The enormous success of the Internet has now made distributed processing and networking everyday realities for many millions of people worldwide. A discussion of the evolution of the Internet as well as its challenges for the future is provided in a paper on Transmission Control Protocol/Internet Protocol by Britton, Tavs, and Bournas,[12] and the entire contents of Volume 37, No 1, 1998 of the *IBM Systems Journal* is devoted to Internet computing.

Database technology

Database technology emerged in the 1960s and 1970s, driven by the needs of transaction and batch environments to minimize the computing resources used per unit of work while allowing granular access to data at the field level rather than at the record level with appropriate access controls and integrity. In the 1980s, however, driven by needs for improved programmer and end-user productivity, the concept of a relational database that was introduced by Codd[13] became the dominant database architecture, with implementations on virtually all commercially available system platforms. The development of the relational database had its

origins in System R, an experimental database management system developed at IBM Research, designed to be unusually easy to use. System R and the structured query language are described in "System R: An Architectural Overview" by Michael Blasgen and coauthors, reprinted here. A discourse by Selinger[14] introduced database technology, traced its evolution, and projected future directions as seen from 1987.

Multiple processor complexes

However dramatically processor speeds increase following Moore's Law,[15] the insatiable demand for computing power has required various ways of lashing together multiple processors in a coordinated fashion to handle a given application or set of applications. Implementations include tightly coupled and loosely coupled processor complexes and a variety of distributed systems networked together. More recently, massively parallel systems have emerged, enabling hugely scalable implementations incorporating hundreds or thousands of processors addressing tasks as diverse as weather forecasting, nuclear explosion simulation, commercial business transactions, and playing chess. IBM's success in this area stems from the introduction of a family of massively parallel systems based on general-purpose processors, connected via a high-speed, nonblocking switch, and specialized operating system management software designed to afford many aspects of a single system image. There are a wide variety of ways to distribute resources across a multitude of processors. "SP2 System Architecture" by Tilak Agerwala and coauthors, reprinted here, not only outlines the architecture utilized in the SP2* system, but also describes the design alternatives and dominant programming models in parallel commercial computing. An alternative approach for implementing a multiple processor complex is utilized in the S/390 Parallel Sysplex Cluster, which is the topic for the entire issue of Volume 36, Number 2, 1997 of the *IBM Systems Journal*.

Summary

The changes in computer systems and architecture since the inception of the *IBM Systems Journal* in 1962 are enormous. Technology and innovation have transformed the computer systems landscape from tremendously expensive devices found in locked glass houses in the bowels of large institutions and accessible only to experts, to flexible systems large and small, found almost everywhere, and accessible to many millions of people. Underlying this transformation are system architectures that have provided durable foundations for continued enhancement and enrichment of systems for an ever growing diversity of applications. We have highlighted as turning points some of the key architectural constructs documented in the *IBM Systems Journal* that have contributed substantially to this evolution.

Cited references

1. G. A. Blaauw and F. P. Brooks, Jr., *Computer Architecture: Concepts and Evolution*, Addison-Wesley Publishing Company, Reading, MA (1997).
2. H. Lorin, "Systems Architecture in Transition—An Overview," *IBM Systems Journal* **25**, Nos. 3/4, 256–273 (1986).
3. G. A. Blaauw and F. P. Brooks, Jr., "The Structure of SYSTEM/360, Part I—Outline of the Logical Structure," *IBM Systems Journal* **3**, Nos. 2&3, 119–135 (1964).
4. J. Fotheringham, "Dynamic Storage Allocation in the Atlas Computer, Including the Use of a Backing Store," *Communications of the ACM* **4**, No. 10, 435–436 (November 1961).
5. M. E. Hopkins, "A Perspective on the 801/Reduced Instruction Set Computer," *IBM Systems Journal* **26**, No. 1, 107–121 (1987).
6. J. H. McFadyen, "Systems Network Architecture: An Overview," *IBM Systems Journal* **15**, No. 1, 4–23 (1976).
7. P. E. Green, "An Introduction to Network Architectures and Protocols," *IBM Systems Journal* **18**, No. 2, 202–222 (1979).
8. J. P. Gray and T. B. McNeill, "SNA Multiple-System Networking," *IBM Systems Journal* **18**, No. 2, 263–297 (1979).
9. J. M. Jaffe, F. H. Moss, and R. A. Weingarten, "SNA Routing: Past, Present, and Possible Future," *IBM Systems Journal* **22**, No. 4, 417–434 (1983).
10. H. Lorin, "Distributed Processing: An Assessment," *IBM Systems Journal* **18**, No. 4, 582–603 (1979).
11. A. L. Scherr, "Structures for Networks of Systems," *IBM Systems Journal* **26**, No. 1, 4–12 (1987).
12. E. G. Britton, J. Tavs, and R. Bournas, "TCP/IP: The Next Generation," *IBM Systems Journal* **34**, No. 3, 452–471 (1995).
13. E. F. Codd, "Relational Model of Data for Large Shared Data Banks," *Communications of the ACM* **13**, No. 6, 377–387 (June 1970).
14. P. G. Selinger, "Database Technology," *IBM Systems Journal* **26**, No. 1, 96–106 (1987).
15. G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics* **38**, No. 8, 114–117 (April 19, 1965).

Alan G. Ganek *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (electronic mail: ganek@us.ibm.com).* Mr. Ganek is Vice President, Technical Strategy and Worldwide Operations, IBM Research, with leadership responsibility for strategic and technology outlook, portfolio management, and Research Division processes. In addition, he oversees operational services supporting the division, including finance, information services, technical journals, and site operations such as facilities management and environmental control. He joined IBM as a software engineer in 1978 in Poughkeepsie, New York, where he was involved in operating system design and development, computer addressing architecture, and parallel systems architecture and design. He was the recipient of IBM Outstanding Innovation Awards for his work on Enterprise Systems Architecture/370 and System/390 Parallel Sysplex Design. He held several management positions in IBM's MVS and VM/XA operating systems development organizations. Mr. Ganek was appointed Director of Worldwide Software Manufacturing Strategy in 1990, and in 1992, he was named Programming Systems Director, Quality and Development Operations, leading quality programs for the Programming Systems Division. He was also responsible for the software development processes including tools, technology, metrics, information development, and software service for IBM's software community. In 1994, he was appointed Director of Solutions Development, IBM Telecommunications and Media Industry Unit, where he was responsible for development activity for IBM's telecommunications and media industry customers worldwide, including regional and inter-exchange carriers, cable and wireless providers, broadcasters, entertainment companies, the sports industry, and publishers. He assumed his current position in January 1998. He holds eight patents. Mr. Ganek received his M.S. degree in computer science from Rutgers University in 1981.

Edward H. Sussenguth *Cary, North Carolina.* Dr. Sussenguth was educated at Harvard University, joined IBM in 1959, and retired at the end of 1990. Among the many positions he held in IBM was Division Director, responsible for the inception and development of Systems Network Architecture (SNA). He has been the author of numerous publications, holds several patents, and has been a keynote speaker in nine countries. Dr. Sussenguth was appointed an IBM Fellow in 1981 and was the first president of the IBM Academy of Technology. He was elected a member of the National Academy of Engineering and is a Fellow of the IEEE (Simon Ramon Medal recipient).

Reprint Order No. G321-5701.