Preface

Over the past decade, computer systems based largely on commodity processors from manufacturers such as AMD, Cyrix, and Intel, commodity operating systems such as Linux, Novell, and Windows, and industry-standard I/O protocols such as Ethernet, PCI, and SCSI, have rapidly moved upmarket into many mission- and performance-critical server applications. What were originally deployed as personal systems a decade ago have matured into oxymoron "PC servers" and are now being used for electronic commerce, web, file, and e-mail serving, enterprise resource planning, business intelligence, databases, messaging, telecommunications, storage, and a host of other server-class applications. The total worldwide server customer revenue for the year 2000 for 32-bit Intel Architecture-based servers was $26 billion, with an annual growth rate of 20%.

Penetrating this market has already required remarkable advances in performance, robustness, and scalability. Yet even higher levels of capability can and should be delivered to our customers, especially in the areas of performance, scalability, cost, and RAS. For example, to continue to participate in the high-scalability SMP server market, PC servers must go beyond their current four- to eight-way SMP scalability limits, to support single operating system images of 64 processors and more. Furthermore, when such a large and expensive system image is supporting a mission-critical application, the reliability of the operating system, middleware, and application software running on that image must be improved relative to today's levels.

In the e-commerce and web-serving customer base, there is a quickly emerging requirement for improved volumetric and power density for these massively parallel applications. Many customers are encountering increasing constraints on floor space and power dissipation, and must furthermore cope with stunningly high workload variability and growth rates. Current PC servers, when packed as densely as possible into racks, are difficult to cool and may not provide the most effective system partitioning. It is possible that alternative packaging and partitioning methods will become popular with these customers.

Finally, we are seeing the need for an increasing number of appliances, or functionally specialized servers. An appliance is a turnkey server that provides a single function extremely well and requires minimal user setup and maintenance; a good example is a web-serving appliance. It is expected that worldwide appliance revenue for web serving, proxy/caching/security, e-mail, workgroup, and file/print and storage applications will grow from $3 billion in 2001 to $12 billion in 2004, with an average annual growth rate of 73%.

This issue of the *IBM Journal of Research and Development* is devoted to advances in PC server development; it contains selected papers that describe potential breakthrough improvements in the central issues of system performance, scalability, price/performance, and software availability. We should note that the PC server industry is an active area of intense innovation, and we can hope to present only a small subset of the important achievements in this industry in any limited space. Thus, we begin with a few papers addressing performance and scalability issues.

The paper by Hu et al. describes the Adaptive Fast Path Architecture (AFPA) software design. AFPA is a software system that substantially improves the performance of web-serving and other applications by using an in-memory cache to serve static content, and a reverse-proxy agent to distribute requests for dynamic content to multiple servers. System efficiency is enhanced by maximizing the number of requests that are handled entirely within the kernel, thus avoiding context-switching overhead whenever possible. AFPA has been demonstrated to double a server's capacity for serving static web content, and it has allowed IBM to establish a leadership position in web-serving performance. It has been ported to Windows, OS/390, AIX, and Linux, and would be a cornerstone technology for a high-performance web-serving appliance.

The paper by Brock et al. describes the construction of a cache-coherent nonuniform memory access (ccNUMA) machine using Intel Standard High Volume (SHV) building blocks, a prime objective of the work being to develop a highly scalable Intel machine with minimal investments in hardware and software. This group has successfully assembled a 16-way ccNUMA system from four 4-way SHV processors, ported a number of operating systems to this machine, and extensively measured and optimized the performance of several computation-intensive workloads. The group used a combined hardware/software approach to support application-level performance tuning, in which a hardware-performance counter card measures the frequency of remote-memory accesses (a dominant source of performance overhead in a NUMA machine) and a software resource set abstraction allows application-level threads to improve performance and scalability by specifying their execution and memory affinity. This hybrid approach was demonstrated and shown to be successful for some computation-intensive workloads. An interesting conclusion from this work is that, for the prototype, performance and scalability were often limited by local-memory bandwidth rather than by the effects of remote-memory access latency.

In the paper by Nanda, the Everest NUMA-protocol controller architecture is described and evaluated. Everest supports high-performance cache coherence and message-passing protocols for partitionable distributed shared-memory systems that use commodity SMPs as building blocks. It performs high-throughput NUMA protocol

handling by using multiple parallel protocol engines, split request–response handling, and a pipelined design. It also has a novel directory structure that uses roughly the same amount of memory as a sparse directory, but retains the benefits of a full-map directory. Many customers purchase highly scalable SMP or NUMA systems, and then want to run multiple independent operating systems on partitions of that machine for error containment, workload consolidation, manageability, and other reasons. For this reason the Everest design also enables partitioned operation, in which each SMP building block supports an independent, isolated operating system instance, and provides hardware facilities for high-performance, secure communications between partitions. The paper presents performance simulation results that show that these Everest features have a positive impact on system-level performance.

IBM has recently announced and made available to the industry Memory Expansion Technology (MXT), an on-the-fly data-compression technology which promises to transform the price/performance equation for PC servers by expanding a system's apparent memory size by a factor of 2 for most applications, at very little additional cost. The next five papers describe various aspects of MXT. We begin with two papers that describe the theory and algorithms that underlie IBM's implementation of on-the-fly memory compression, and follow with three papers that describe a commercial implementation of MXT, assess its software support and performance impact, and describe its competitive implications in the marketplace.

The first paper, by Franaszek et al., provides an overview of algorithms and data structures for compressing and decompressing data as it flows into and out of memory. These algorithms are unique in that they work effectively on the relatively small fixed-size cache lines favored by processors, and are suitable for high-speed hardware implementation in a processor-memory cache hierarchy. Furthermore, as variable-sized compressed data flows into memory, the data blocks are stored in such a manner as to minimize overheads due to directory size and storage fragmentation. Compressed data is by its nature variable in size, so totality of the data stored in main memory with MXT will vary over time. To make the most of the additional resources provided by MXT, a system should include facilities and services to efficiently manage this variability. The requirements and opportunities of such management are discussed in the paper.

The second paper, by Franaszek and Robinson, discusses research on compressed-memory algorithms and architecture that was performed before and during the design of the MXT architecture. This work is fundamental to the design of the MXT compressed-memory organization, which is described in the papers that follow. The paper discusses and analyzes three problems that arise in the design of compressed-memory systems: allocating and managing storage for variable-sized compressed blocks, choosing a static versus a dynamic directory structure, and selecting a smaller versus a larger block size. This paper quantitatively analyzes these tradeoffs and makes design recommendations using analytical methods and simulations.

The next three MXT papers describe an implementation of data compression in a commercial Intel-compatible chipset, the software support and performance impact of data compression, and its possible competitive impact. The first paper, by Tremaine et al., describes the architecture and implementation of a memory-controller chip that supports a four-way Pentium III SMP. This chip supports a large L3 cache to hide the latency of decompression, and contains high-speed data compression and decompression circuitry that implements the algorithms described in the previous papers. The chipset also incorporates memory-management hardware that dynamically allocates main-memory storage in small sectors to accommodate storing the variable-sized compressed data without the need for garbage collection or significant fragmentation. This chip is the first commercially available memory controller to employ real-time memory data compression. This "Pinnacle" chipset was developed jointly with Serverworks, a division of Broadcom Inc., and will be available to the industry through them.

The next paper, by Abali et al., describes the MXT memory resource management services which have been developed to support the Microsoft Windows 2000 and Linux 2.2 and 2.4 operating systems. Specifically, the dynamic compressibility of the data stored in memory is monitored, and the amount of real memory available to the operating system is adjusted to permit as many pages of memory as possible to remain in memory, given the compressibility of those pages. The paper also provides an empirical measurement of the performance impact of data compression, and shows that under the SPEC CPU 2000 workload, there is negligible performance degradation. The paper concludes with empirical measurements of the compressibility of memory for the SPEC CPU 2000, Synopsis, Photoshop, MSDN Install, and the contents of certain well-known web sites, and shows that a compression ratio of 2:1 is routinely achievable.

The last MXT paper, by Smith et al., quantifies the impact of memory compression on price/performance. Assuming a 2:1 compression ratio, for "price twins" (i.e., two systems having the same price, one with data compression and one without), MXT can be used to double the amount of apparent memory and thus achieve a performance improvement. For "performance twins" (i.e., two systems having the same amount of apparent

memory, one with data compression and one without) MXT can provide an equivalent amount of apparent memory at half the memory price. Depending on which approach is taken and on the workload, MXT is shown to improve price/performance by 25% to 70%. In the PC server market, this can be equivalent to the entire gross margin for many of these machines.

Our final paper, by Castelli et al., describes an approach for improving software availability, when one cannot improve the quality of the underlying code, by proactive detection and management of software aging. Essentially, if a system exhibits software aging and can tolerate periodic restarts or "rejuvenations," judicious rejuvenation can substantially improve system availability, especially in a high-availability cluster environment. The authors describe the general problem of software aging, present an analytical model for software aging and rejuvenation, and describe a software product that is designed to detect software aging and proactively perform rejuvenations. The analytical results indicate that the probability that the cluster is unavailable due to software-aging-induced outages can be reduced by 25% to 90%, depending on the rejuvenation policy.

We would like to thank the many authors from the IBM Thomas J. Watson Research Center and the IBM xSeries Division who have taken the time to prepare these excellent papers. Thanks are also due the reviewers, who contributed substantially to the quality of the papers.

Richard E. Harper

Guest Editor