IBM Memory eXpansion Technology (MXT) Debuts in a ServerWorks Northbridge

Sujith Arramreddy[Ψ], David Har, Kwok-Ken Mak, T. Basil Smith, R. Brett Tremaine[γ], Mike Wazlowski

[Ψ]ServerWorks Inc
2251 Lawson Lane
Santa Clara, CA 95054

IBM T. J. Watson Research Center
P.O. Box 218
Yorktown, NY 10598

[γ]Email: afton@us.ibm.com
Phone: 914-945-2710
FAX: 914-945-2141

Abstract

Several state-of-the-art technologies are leveraged to produce a low-cost and high performance single chip memory controller with a new memory system architecture, more than doubling the effective size of the installed main memory without significant added cost. This unique chip is the first of its kind ever to employ real-time main memory content compression at a performance competitive with the best the market has to offer. A large low-latency shared cache exists between the Intel PIII processor bus and a content compressed main memory. Novel high-speed and low-latency on-chip hardware performs real-time compression and decompression of data traffic between the shared cache and the main memory. Sophisticated memory management hardware dynamically allocates main memory storage in small sectors to accommodate storing the variable sized compressed data, without the need for "garbage" collection or significant wasted space due to fragmentation. Though the main memory compression ratio is limited to the range 1:1 - 64:1, typical ratios range between 2:1 - 6:1, as measured in "real-world" system applications. The 133MHz memory controller chip is fabricated in 0.25 micron standard cell ASIC technology and is packaged in a 731 EPBGA. The chip, called Pinnacle, is marketed by ServerWorks, Inc, in Santa Clara, CA.

1.0 Introduction

Memory costs dominate both large memory servers and expansive compute server environments, like those employed in today's "data centers" and "compute farms". These costs are both fiscal and physical (e.g., volume, power, and performance associated with the memory system implementation), and often aggregate to a significant cost constraint that the Information Technology (IT) professional must tradeoff against compute goals.

Data compression techniques are employed pervasively through the computer industry to increase the overall cost efficiency of storage and communication media. However, despite some experimental work [7][3], system main memory compression has not been exploited to its potential. IBM's Memory eXpansion Technology (MXT) addresses the system memory cost issue head-on with a new memory system architecture that more than doubles the effective capacity of the installed main memory without significant added cost.

Engineers from IBM and ServerWorks jointly developed a new Intel PentiumIII/Xeon Bus compatible, low cost single chip memory controller (north bridge) with MXT, called "Pinnacle". This unique chip is the first commercially available memory controller to employ real-time main memory content compression at a performance competitive with the markets best products. The Pinnacle chip is ServerWorks' flagship product for the commercial server market.

2.0 Architecture

Conventional "commodity" computer systems are designed with a common architecture, where a collection of processors are connected to a common SDRAM based main memory through a memory controller chip set. We chose a two-level main memory architecture [1] shown in Figure 1, consisting of a large shared cache coupled with a typical main memory array. The high speed cache, containing the frequently referenced processor data, architecturally isolates the overall system performance from access latency to the main memory; thereby opening opportunities for trading off increased memory access latency for greater function. For example, remote/distributed, very large and/or highly reliable features may be incorporated without adverse effects to overall system performance.

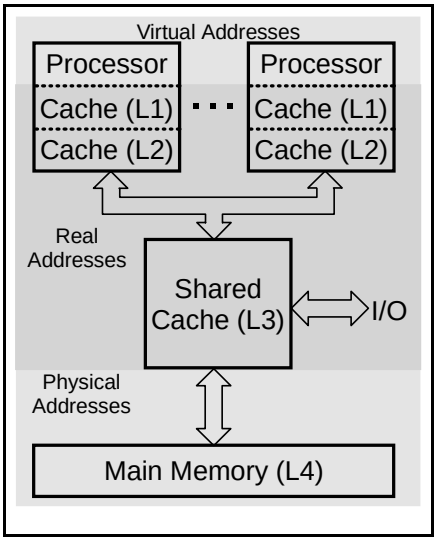


Figure 1: System Memory Hierarchy.

The shared cache, coupled with the recent advent of high density 0.25 micron and smaller ASIC technology, is leveraged to incorporate a new "compressed" main memory architecture. Special logic intensive compressor and decompressor hardware engines provide the means to simultaneously compress and decompress data as it is moved between the shared cache and the main memory. The compressor encodes 1KB data blocks into as compact a result as the algorithm permits. A special memory management architecture is employed to permit storing the variable sized compressed data units in main memory, while mitigating fragmentation effects and avoiding "garbage collection" schemes. This new architecture serves to halve the main memory cost, without any significant degradation in overall system performance.

3.0 Pinnacle Chip

The Pinnacle chip host bridge controller provides connectivity between 1-4 Intel Pentium III/Xeon processors, a 32MB DDR SDRAM shared cache memory, main memory and up to two independent remote IO PCI bridge (CIOB) chips. The low cost single chip controller is optimized for use in a wide range of high performance server system applications. The full featured (below) chip may be used in two primary memory configurations as shown in Figure 2.

- **Single ChipController**
  - 731 (525 signal) EPGA 45x45 mm package
  - Single 2.5VDC power supply (6 Watt typical)
  - 100/133 MHz operation, (synchronous to a processor, cache and memory interfaces)
- **Processor Interface**
  - Pentium III/Xeon; 1-4 processors, 36-bit address
  - 8 Entry request queue
  - 8x32B burst write buffer and 8x4B write buffer
- **IO Interface**
  - Dual independent full duplex 533MB/s remote IO bridge Inter-Module Bus links (IMB)
  - 16 Entry request queue per link
- **Shared Cache**
  - Dual ported on chip directory supports 16GB "real" address space
  - 32MB DDR SDRAM cache with 1KB line, 4-way set associative, ECC protected
  - Reference state for snoop filtering (256B granularity and IO)
  - 1.6GB/s-2.1GB/s DDR SDRAM cache interface
  - LRU replacement, write-back and write allocate policies
- **Main Memory**
  - 1.6GB/s-2.1GB/s access to 16GB uncompressed or 8GB (16GB effective) compressed memory
  - Supports PC100/PC133 DIMM's
  - ECC protected w/scrub for x4 and x8 "chip kill"
  - Hardware memory management with hardware accelerated 4KB page move, swap and clear
  - Real-time hardware dataflow compression decompression for 1:1 - 64:1 main memory content compression
- **Hardware performance monitor facility**
- **I2C bus access to all internal registers**

The chip internal structure, shown in Figure 3, is designed with all primary internal data flow at 128-bit wide, pipelined and full duplex. The cache and IMB interface operate at double data rate with respect to the chip clock. Any processor or IO memory references are directed to the cache controller, resulting in cache directory look-up to determine if the address is

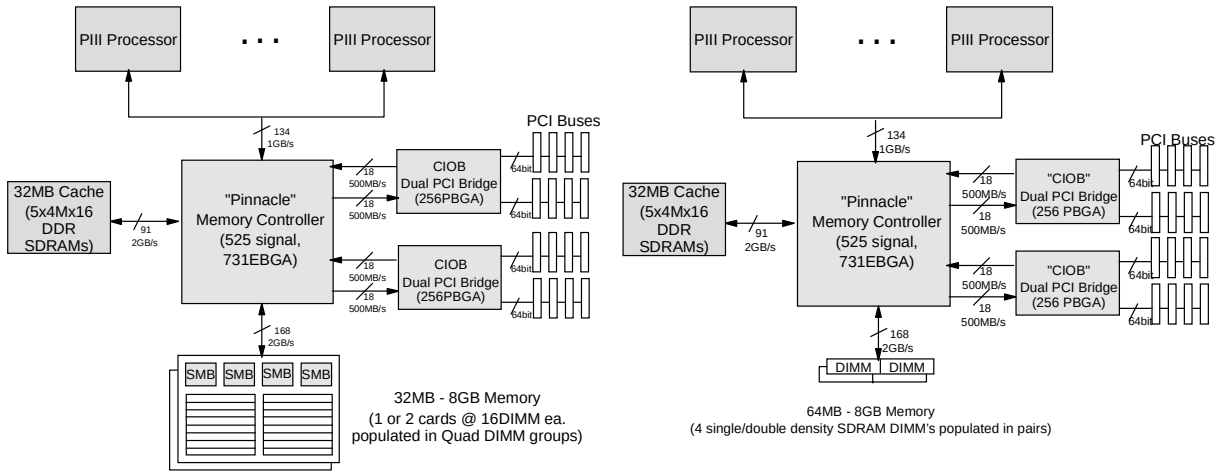


Figure 2: Pinnacle chips configurations.

contained within the cache or not. Cached references are serviced directly from the cache, while cache read misses are "deferred", and the least recently used cache line is selected for replacement with the new cache line that contains the requested address. The cache controller issues a request for the new cache line from the main memory controller, while at the same time writing back the old cache line to the write back buffer (wtq) in cases where the old cache line contains modified data.

Meanwhile, the memory controller first reads a small address translation table entry from memory to determine the location of the requested data. Then the memory controller commences reading the requested data. Data is either streamed around the decompressor (decomp) when uncompressed, or through the decompressor when compressed. In either case, the data is then streamed through the elastic buffer (rdq) to the cache. The memory controller provides 7 cycle advance notification of when the requested 32B data will be in the critical word buffer (cw). This permits the processor bus controller to arbitrate for a deferred read reply, and deliver data without delay.

Cache write back activity is processed in parallel with read activity. Once an entire cache line is queued in the write back buffer (wtq), the compression commences and runs uninterrupted until complete, 256 cycles later. Then the memory controller stores the compressed data when a spatial advantage exists, otherwise the memory controller stores the write back data directly from the write back buffer. In either case, the memory controller must first read the translation table entry for the write back address to allocate the appropriate storage and update the entry accordingly, before writing it back to memory. The data itself is then written to memory, within the allocated sectors.

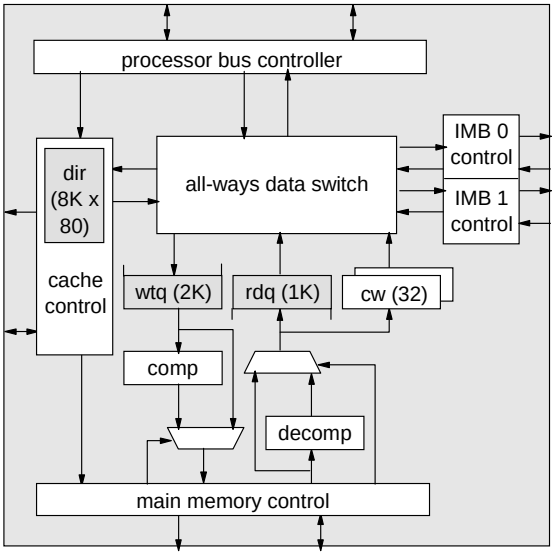


Figure 3: Pinnacle chip block diagram.

3.1 Shared Cache Subsystem

The shared cache provides low latency processor and IO subsystem access to frequently accessed uncompressed data. The data/code/IO unified cache content is always uncompressed and accessed at 32B granularity. Write accesses smaller that 32B require the cache controller to perform a read-modify-write operation for the requesting agent.

The 32MB, 4-way set associative cache is organized into four banks of 8K x 1024B lines, or 8K sets of four lines each. Cache lines within a set are replaced according to the least recently used (LRU)

policy. The cache line size was optimally chosen to minimize the size of the on chip cache directory as well as to be equivalent in size to the compression algorithm block size.

The shared cache directory contains a unique 17-bit entry, organized as shown below, for each of the 32K cache lines. The tag address bits permit caching the low order 16GB of real address space. The directory is implemented on chip as a 8K x 76 bit dual port (1 read and 1 write) SRAM. The four entries associated with a set are accessed concurrently along with associated LRU state and parity bits.

16:13	12	11	10:0
PR<3:0>	IOR	M	Tag<10:0>

The relatively long (1KB) cache line merits special design consideration. For example, the processor reference bits (PR<3:0>) are used to mitigate extraneous cache coherency snoop traffic on the processor bus. These four bits are used to indicate when any processor has referenced any one or more quarter cache line (256B) segments. When a cache line is evicted, only "referenced" 256B cache line segments, verses the entire 1KB line, need be invalidated on the processor bus.

Shuttling the wide lines in and out of the cache during cache line replacement requires at least 64 system clock cycles, or 32 accesses for each write back and line fill operation. To alleviate processor access stalls during the lengthy cache line replacement, the cache controller permits two logical cache lines to coexist within one physical cache line. This mechanism permits the cache line to be written back, reloaded, and referenced simultaneously during cache line replacement.

During replacement, a state vector is maintained to indicate *old*, *new*, or *invalid* state for each of the thirty-two, 32B sub-cache lines within the physical line. As 32B sub-cache lines are invalidated or moved from the cache to the write back buffer, the sectors are marked *invalid*, indicating that the associated new sub-cache line may be written into the cache. Each time a new sub-cache line is loaded, the associated state is marked *new*, indicating that processor/IO access is permitted to the new cache line address. Further, processor/IO accesses to the old cache address are also permitted, when the associated sub-cache lines are marked *old*. Cache lines are always optimally fetched and filled, such that the sub-cache line write back follows the same sub-cache line order to maximize the amount of valid cache line at all times.

The cache can support up to two concurrent cache line replacements. Two independent 1KB write back buffers (wtq) facilitate a store-and-forward pipeline to the main memory, and one 1KB elastic buffer (rdq) queues line fill data when the cache is unavailable for access. A write back buffer must contain the entire cache line before the main memory compressor may commence the compression operation. Conversely, the line fill data stream is delivered directly to the cache as soon as a minimum 32B granule of data is contained within the buffer. Two independent 32B *critical word* (CW) buffers are used to capture the data associated with cache misses for direct processor bus access.

3.2 Main Memory Subsystem

The main memory subsystem stores and retrieves 1KB cache lines in response to shared cache write back (write) and line fill (read) requests. Data is stored within a 32MB-16GB array comprised of industry standard PC100/PC133 SDRAM DIMM's. The memory controller supports two separate DIMM configurations for optimal application in both large and small server applications. The *direct attach* configuration supports 2 or 4 single/double density DIMM's directly connected to the Pinnacle chip without any glue logic. Whereas the large memory configuration supports 1 or 2 cards with synchronous memory buffer (SMB) chips and 16 DIMM's each, populated in quad-DIMM groups. In either configuration, the array is accessed via the 144-bit (16B + ECC) data interface with 32B - 256B access granularity. For minimal latency, uncompressed data references are always retrieved with the critical 32B first and 256B address wrapped as shown in Figure 4.

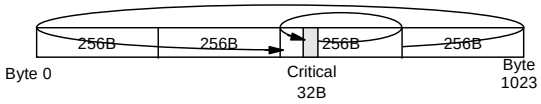


Figure 4: Critical word fetch order.

The main memory subsystem may be configured to operate with compression disabled, enabled for specific address ranges, or completely enabled. When compression is disabled, the physical memory address space is directly mapped to the real address space in a manner equivalent to conventional memory systems. Otherwise, the memory controller provides real to physical address translation to accommodate dynamically allocating storage for the variable size data associated with compressed 1KB lines. The additional level of address translation is carried out completely in hardware using a translation table apportioned from the main memory.

The physical memory is partitioned into two regions, or optionally three when uncompressed memory is configured. The memory is comprised of two primary data structures; the *sector translation*

table (STT) and the *sectored memory* as shown in Figure 5. The STT consists of an array of 16B entries, where each entry is directly mapped to a corresponding 1KB real address. Therefore, the number of STT entries is directly proportional (1/64) to the size of the real address space[1] declared for a given system. Each entry describes the attributes for the data stored in the physical memory and associated with the corresponding 1KB address. Data may occur in one of three conditions:

- Compressed to <=120 bits
- Compressed to > 120 bits
- Uncompressed

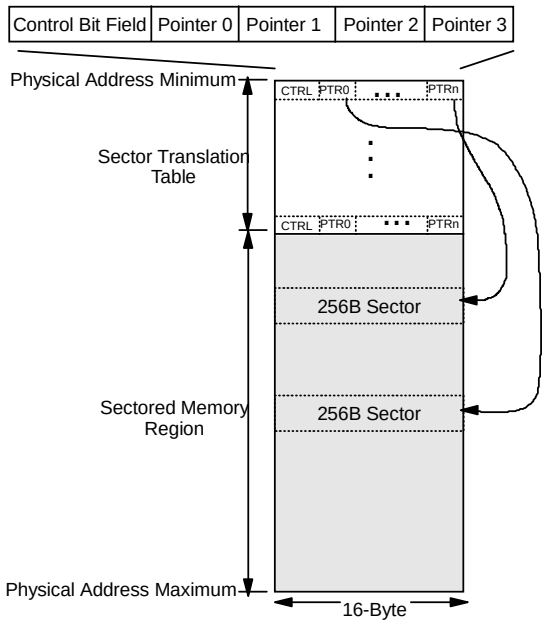


Figure 5: Memory organization.

When a 1KB data block is compressible to <= 120 bits, then the data is stored directly into the STT entry with appropriate flags, yielding a maximum compressibility of 64:1. Otherwise, the data is stored outside the entry in 1-4, 256B sectors, with the sector pointers contained within the STT entry. For the case where the data block is uncompressed, four sectors are used and the STT entry control field indicates the "uncompressed" attribute. In cases where unused "fragments" of sector memory exist within a 4KB real page, any new storage activity within the same page can share a partially used sector in increments of 32B. A maximum of two 1KB blocks within a page may share a sector. This simple 2-way sharing scheme typically improves the overall compression efficiency by 15%, nearly all the potential gain attainable from combining fragments by any degree of sharing [6].

The sectored memory consists of a "sea" of 256B chunks of storage or sectors, that are allocated from a "heap" of free sectors available within the sectored memory region. The heap is organized as a linked list of unused sector addresses, with the list head maintained within a hardware register. The list itself is stored within the free sectors, so the utilization of sectors oscillates between holding the free list and data. As shown in Figure 6, each node of the free list contains pointers to sixty-three free 256B sectors and one pointer to the next 256B node in the free-list. Since the node is itself a free or unused 256B sector, effectively the free-list requires no additional storage.

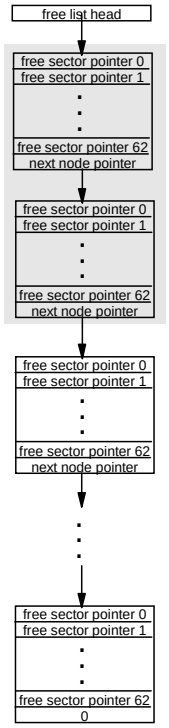


Figure 6: Free list.

A small hardware cache contains the leading two nodes (shaded in Figure 6) of the free list, for rapid access during allocation and deallocation of sectors associated with data storage requests.

Uncompressed memory regions are areas of the real address space in a compressed memory in which the data is never compressed. The Pinnacle chip supports 0-4 such regions, where each region is configurable as a 32KB-256MB range, 32KB aligned. These are apportioned from the top of the sectored memory and are direct mapped as shown in Figure 7. The access latency to these regions is minimized as data is directly addressable without the intermediate step of referencing a STT entry. And of course, the data is fetched with the requested 32B first, as is always the case for uncompressed data.

The regions within the STT that contain entries for addresses within unsectored regions are never referenced. Not to wasted, these "holes" within the STT are made available as additional sectored storage through incorporation on to the free list.

[1] The real address space is defined to the operating environment through a hardware register. The BIOS firmware initializes the register with a value based on the quantity and type of DIMM's installed in a system. When compression is enabled, the BIOS doubles this value to indicate a real address space twice as large as is populated with DIMM's.

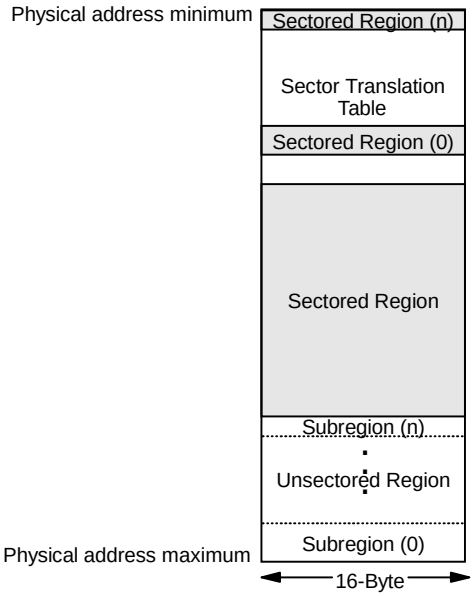


Figure 7: Unsectored memory organization.

3.3 Page Operations

A beneficial side effect of "virtualizing" the main memory through a translation table, is that simple alteration of the table entry can be used to relocate and clear data associated with a given table entry. We capitalized on this notion by implementing a programmed control mechanism to enable memory page (4KB) manipulation, at speeds ranging between 0.1-3.0 micro seconds, depending on the amount of processor bus coherency traffic required. The Pinnacle chip supports page swap, clear, invalidate, and flush-invalidate operations.

3.4 Compression/Decompression

The compression/decompression mechanism is the cornerstone of MXT. Compression, as applied in the main memory data flow application, requires low latency and high bandwidth in the read path, and of course it must be loss-less. Although a plethora of compression algorithms exist, none met our architectural criteria. We chose to leverage the recently available high density (0.25 micron) CMOS ASIC technology by implementing a gate intensive, parallelized derivative [2] of the popular Ziv-Lempel (LZ77) "adaptive dictionary" approach. With this new scheme, the unencoded data block is partitioned into *n* equal parts, each operated on by an independent compression engine, but with shared dictionaries. It has been shown experimentally, that parallel compressors with cooperatively constructed dictionaries have essentially equivalent compression efficiency as the sequential LZ77 method [2].

The Pinnacle chip embodiment contains four compression engines, each operating on 256B (¼ of the 1KB uncompressed data block), at the rate of 1B/cycle, yielding a 4B/cycle aggregate compression rate. Referring to Figure 8, each engine contains a history buffer or *dictionary* consisting of a 255-byte Content Addressable Memory (CAM) that functions as a shift register. Attached to each dictionary are four 255-byte (4080) comparators for locating the incoming reference byte within the entire dictionary structure. Each clock cycle, one byte from each 256B source data block (read from the shared cache write back buffer) is simultaneously shifted into a respective dictionary and compared to the accumulated (valid) dictionary bytes. The longest match of two or more bytes constitutes a *working string* while the copy in the dictionary is the *reference string.* Should a single byte match or no match be found, as may be the case for random data, the reference byte is a *raw character*.

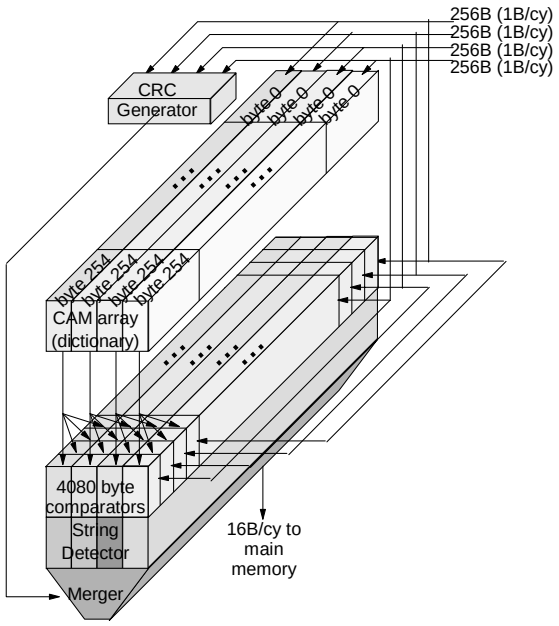


Figure 8: Compressor block diagram.

Compression occurs when *working strings* within the compare data stream are replaced with location and length encoding to the *reference strings* within the dictionary. However, it can be seen in the table below, that the encoding scheme may result in a 256B uncompressed data stream actually expanding to 288 bytes for a given engine. Therefore special detection logic is employed to detect when the accumulated aggregate compressed output exceeds 1KB (or a programmed threshold), causing compression to be

aborted and the uncompressed data block stored in memory.

Compressed Data Type	Encoding
Raw Character	{0, data byte}
String	{1, primary length, position, secondary length}

Strings are detected by one of 255 detectors from any one of the four dictionaries. Once an emerging string is detected, future potential strings are ignored until the end of the current string is detected. The string detector calculates the length and position of a working string. At the end, only the longest string, or that starting nearest the beginning of the dictionary for multiple strings with the same length, is selected. The length field ranges from 2-12 bits to encode the number of bytes in the *working string*, using a Huffman coding scheme. The position field ranges from 2-10 bits to encode the starting address of the *reference string*. Merge logic is responsible for packing the variable length bits stream into a word addressable buffer.

The much simpler decompressor is comprised of four engines the each decoding the encoded compressed data block. Each engine can produce 2B/cycle, yielding an agregate 8B/cy when 1X clocked or 16B/cycle when 2X clocked, as occurs in Pinnacle.

4.0 Reliability-Availability-Servicability

The importance customers place on the RAS characteristics of server-class computers compels server manufacturers to attain the highest cost effective RAS. Main memory compression adds a new facet to this endeavor [4], with the primary goal of detecting any data corruption within the system. Toward that end, the Pinnacle chip includes many RAS specific features (delineated below), with appropriate logging and programmable interrupt control:

- Shared cache ECC
- Shared cache directory parity checking and programmable hardware assisted test
- Main memory ECC with programmable scrub
- IO channel "dead-man" timer time-out recovery
- Hardware configuration/control register write protection with service processor override via I2C bus
- Processor/cache/IO interface protocol checking
- Sector Translation Table entry parity checking
- Sector free list parity checking
- Sector out of range checking
- Sectored memory overrun detection
- Sectors used threshold detection (2)
- Compressor/decompressor validity checking
- Compressed memory CRC protection

Since the compression and decompression functions effectively encode and decode the system data, any malfunction during the processes can produce seemingly correct, yet corrupted output. Further, the hardware function implementation requires a prodigious quantity of (order 1 million) logic gates. Thus, the probability for a logic upset induced data corruption that goes undetected is significant. Although special fault detection mechanisms within the compression/decompression hardware have been incorporated, they cannot provide complete fault coverage. Therefore, we needed an improved method of data integrity protection to minimize the potential for corrupted data to persist in the system without detection.

We employed a standard 32-bit cyclic redundancy code (CRC) computation over the uncompressed data block as it streams into the compressor. When the compression is complete, and the data is to be stored in the compressed format (i.e., the data is compressible, such that a spatial advantage exists over storing the data in the uncompressed format), the check code is appended to the end of the compressed data block, and the associated block size is increased by 4 bytes. Information that is stored in the uncompressed format gains little benefit from the CRC protection as it bypasses the compressor and decompressor functions, and hence is not covered by the CRC protection. Servicing a compressed memory read request results in the decompression of the compressed block and concurrent recomputation of the CRC over the uncompressed data stream from the decompressor. Upon completion of the decompression, the appended CRC is compared to the re-computed CRC. When the two are not equal, an uncorrectable error is signaled within the system to alert the operating system to the event.

5.0 Operating System Software

All commercial computer operating system (OS) software environments manage the hardware memory as a shared resource to multiple processes. In cases where the memory resource becomes limited, (i.e., processes request more memory that is physically available within the machine,) the OS can take steps for continued system operation. Typically, the OS migrates underutilized memory pages (4KB) to disk, and then reallocates the memory to the requesting processes. In this manner, the main memory is used like a cache that is backed by a large disk based storage. This scheme works quite well because the absolute amount of memory is known to the OS. This algorithm still applies to MXT based systems as well.

Although current "shrink wrap" OS software can be used on an MXT machine, the software cannot yet distinguish an MXT based machine from a conventional memory hardware environment. This poses a problem, as the amount of memory known to the OS is twice what actually exists within an MXT machine. Further, the OS is not aware of the notion of compression ratio either. Therefore, the OS cannot detect conditions when the physical memory may be over utilized (i.e., there are too few unused or free sectors left in the sectored memory), and therefore may not invoke the paging management software to handle the situation, possibly leading to a system failure. This condition can occur when the OS has fully allocated the available memory and the overall compression ratio has fallen below 2:1.

Fortunately, minor changes in the OS kernel virtual memory manager are sufficient to make the OS "MXT aware" [5], and these changes are underway in most of the commercial OS products. The same objective can also be accomplished outside the kernel, for example in a device driver or service, albeit less efficiently. We currently have machines deployed with Linux and the Microsoft Windows 2000/NT 4.0 operating systems.

6.0 Performance

MXT compression performance fundamentally ranges between 1:0.98 (1:1) and 64:1, including translation table memory overhead. Figure 10 shows a representative sampling of the many memory content compressibility measurements we have taken from several types of machines. We can take measurements by either, direct measurement on an MXT machine, indirect measurement via a monitor program running on a non-MXT machine, and of course post analysis of memory dumps. Compressibility only drops below 2:1 in the rare case where the majority of the system memory contains random or pre-compressed data.

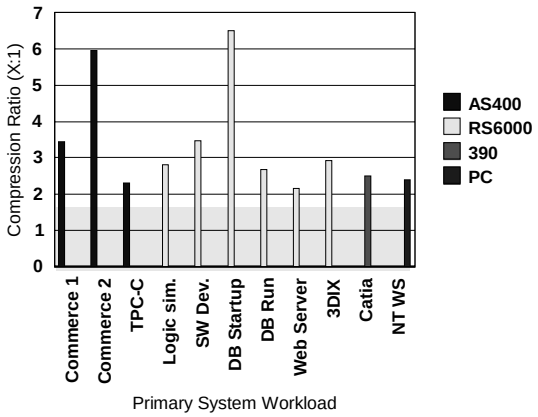
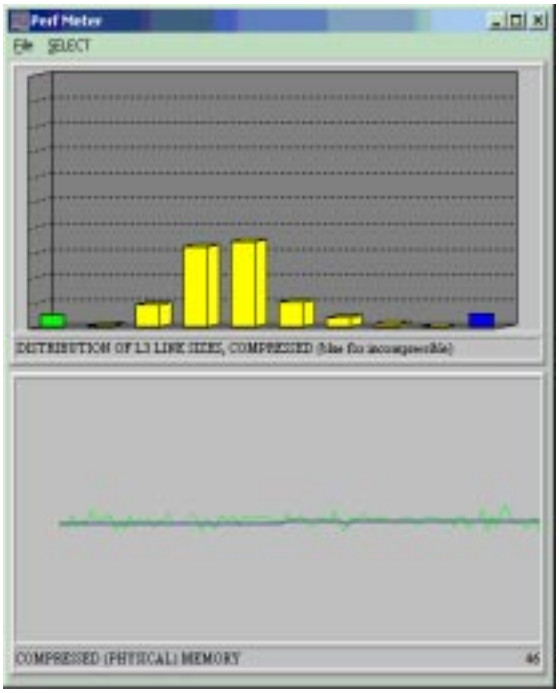


Figure 10: Memory compressibility.

We have observed that the compression ratio of a given machine tends to remain relatively constant throughout the operation of the application set. For example, monitoring the IBM Internet online ordering web server[2] over a period of 10 hours, indicated a compression ratio of 2.15:1 +/- 1%. Further, it can be seen in Figure 11 that the distribution of compressibility is normal. Each bar of the histogram represents the degree of compressibility, where the right most bar is incompressible (1:1), and the left lost bar is maximally compressed (64:1). The lower line represents the degree of change in compressibility over the measurement period.



[2] Specific shadow servers 9q, 9w for web site: http://www.pc.ibm.com/ibm_us/

Figure 9: IBM web site compression distribution.

MXT system performance evaluation can be considered from two primary perspectives; one being the intrinsic performance like that measured on any conventional system, and the other being cost-performance in memory starved applications. Much has been written about the performance benefit additional memory can provide for memory intensive applications. As one might expect, this is where MXT systems really stand out. So much so that we typically see customers experiencing 50%-100% improvement in system throughput. For example, one customer operating a compute farm with several thousand dual processor servers, each containing 1GB of memory, was able to run one job per unit time on each machine. When an equivalent (dual processor and 1GB memory) MXT machine was used in the environment, two jobs could be run concurrently over the same period of time because the 1GB was effectively doubled to 2GB through MXT expansion. Similar memory dependent performance is observed with the behavior of the well known SPECweb99 benchmark. For this case (Figure 11), increasing memory from 256MB to 512MB yields a 45% performance improvement, beyond 512MB the benefit diminishes.

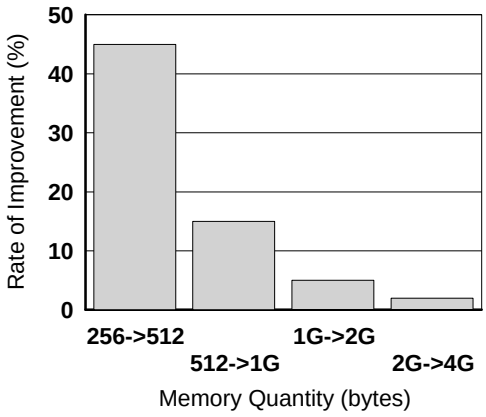


Figure 11: SPECweb99 performance (amount of simultaneous connections) dependence on available system memory.

We began this project with the primary goal of gaining the benefit of doubling the system memory at a negligible cost, but without degrading the system performance. To that end, the Pinnacle chip performance is based on the intrinsic chip reaction times, delineated in the table below, as well as the shared cache hit rate.

Processor Request (with respect to shared cache hit/miss)	**Latency** (bus clock cycles)
Write	6 1 1 1
Read, hit (SDRAM row open and row hit)	8 1 1 1
Read, hit (SDRAM row open and row miss)	14 1 1 1
Read, hit (SDRAM row closed)	14 1 1 1
Read, hit (SDRAM autoprecharge mode)	12 1 1 1
Read, miss compressed (avg)	69 1 1 1
Read, miss not compressed	24 1 1 1
Read, miss compression off	16 1 1 1

The shared cache hit rate is application dependent, and as expected, we typically measure the cache hit rate at roughly 98% on most applications. However, the cache hit rate for large data base applications like TPC-C, SAP and particularly Lotus Notes can range as low as 94%, as measured by quad processor trace driven performance models. These applications tend to reference some database records infrequently, resulting in a prefetch advantage with the long cache line, but little reuse of the data within the line.

Our comparison of MXT system performance with that of a high performance contemporary system, resulted with the two systems having essentially equivalent (within 1 point) performance for the SPECint2000 benchmark. Both machines were IBM 1U commercial servers with 512MB and Intel 733MHz PIII processors, executing program code from the same disk drive. The two systems differ only in the memory controller used. While the MXT system used the ServerWorks Pinnacle chip, the other system used the ServerWorks CNB30LE chip.

MXT provides a system cost leverage not seen since the invention of DRAM. Figure 12 illustrates the degree of this leverage with a case in point. The graph shows how the cost-performance metric for a family of conventional machines (dashed line) is dramatically improved when the effects of MXT are factored in (solid line). We configured a ProLiant DL360 commercial server on the Compaq Inc. Internet web site[3] for retail equipment sales. This server was priced at $13,081. Using the same site to configure a hypothetical MXT equivalent system with half the memory, yielded over 40% savings at $7,103. Viewed another way, a hypothetical MXT "better" system can be configured at the equivalent price to the reference machine. Either way, an MXT system cost-performance metric compares quite favorably with any conventional system.

[3] Referenced on September 11, 1999 at web site: http://www5.compaq.com/products/servers/platforms

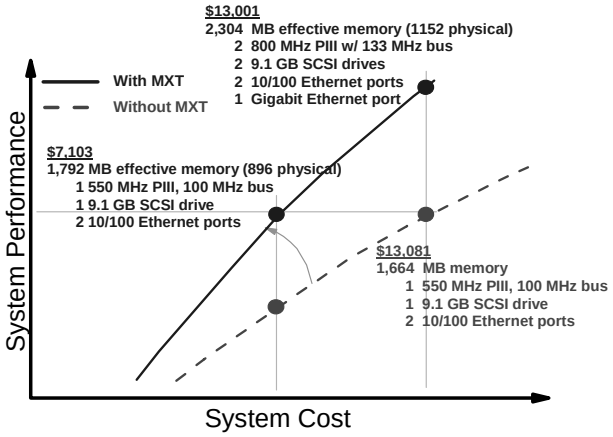


Figure 12: System cost comparison.

MXT is a logical step in the pervasion of compression technologies. MXT is a proven technology that empowers customers to efficiently utilize their memory investment. Information Technology professionals can routinely save $1000's on systems ranging from high density servers to large memory enterprise servers. ServerWorks began sampling the Pinnacle chip in August, 2000, with production availability in the first quarter, 2001. We expect MXT technology to expand its presence into other processor memory controllers, as well as other memory intensive system applications including, disk storage controllers, laptops, and data appliances.

Acknowledgments

MXT and its embodiment in the Pinnacle chip involved significant contributions from many other people. We especially thank: Peter Franaszek and John Robinson for their work on the compression approach; Dan Poff, Rob Saccone and Bulent Abali for operating system and performance measurement work; Michel Hack and Chuck Schulz for their work on the data storage and page operations.

References

[1] M. Abbott, D. Har, L. Herger, M. Kauffmann, K. Mak, J. Murdock, C. Schulz, T. B. Smith, B. Tremaine, D. Yeh, L. Wong *Durable Memory RS/6000 System Design*, Digest of Papers, The 24th Annual International Symposium on Fault-Tolerant Computing, Austin, Texas June 15-17, pp. 414-423, 1994.

[2] P. A. Franaszek, J. Robinson, J. Thomas, Parallel compression with cooperative dictionary construction. In *Proceedings DCC '96 Data Compression Conference*, pp. 200-209, IEEE, 1996

[3] Kjelso, M.,Gooch, M. And Jones, S. Design and performance of a main memory hardware data compressor. In *Proceedings of the 22nd EUROMICRO Conf.*, pp. 423-430, IEEE, 1996.

[4] C. L. Chen, D. Har, K. Mak, C. Schulz, B. Tremaine, M. Wazlowski, "Reliability - Availability - Serviceability of a Compressed Memory System," In *Proceedings of the International Symposium on Dependable Systems and Networks*, pp. 163-168, IEEE, 2000.

[5] B. Abali and H. Franke, "Operating System Support for Fast Hardware Compression of Main Memory Contents," Proceedings of the 27th International Symposium on Computer Architecture, June, 2000.

[6] P. A. Franaszek, J. Robinson, Design and Analysis of Internal Organizations for Compressed Random Access Memories, Research Report RC 21146(94535)20OCT1998, IBM Research, March 1992.

[7] Hovis, et all "Compression Architecture for system memory application", United States Patent US5812817, issued September 22, 1998.