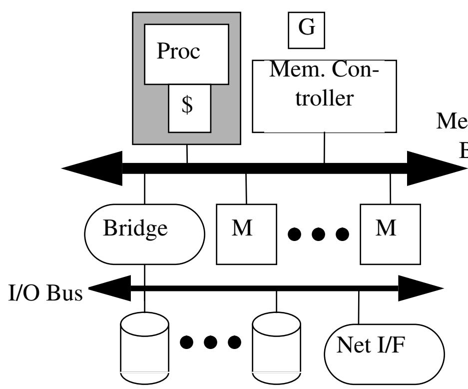
IRAM and SmartSIMM: Overcoming the I/O Bus Bottleneck[1]

Kimberly Keeton, Remzi Arpaci-Dusseau, and David A. Patterson
{kkeeton,remzi,patterson}@cs.berkeley.edu
http://iram.cs.berkeley.edu
Computer Science Division
University of California at Berkeley
387 Soda Hall #1776
Berkeley, CA 94720-1776

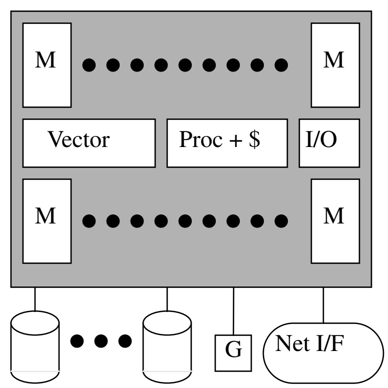
1 Introduction

Internally, a single IRAM (Intelligent RAM) chip presents enormous computing potential due to the high bandwidth and low latency access to memory afforded by tighter integration [14]. While a single IRAM may provide a complete solution in some application domains, one fundamental issue must be addressed in designing more conventional systems with IRAM as the core building block: I/O.

How should an IRAM interact with off-chip I/O devices? As shown in Figure 1(a), conventional computer architectures connect I/O devices to parallel I/O busses, which communicate with memory over a bridge to a parallel memory bus. However, parallel busses are not a natural match for IRAMs, for two reasons. First, the performance of such busses often lags the performance of other components in the computing system. For example, in [1], researchers found that a relatively modern UltraSPARC I workstation is handcuffed by the modest performance of the S-Bus. Second, these conventional parallel I/O busses typically have high pin requirements. As more bandwidth is needed, busses often simply double the number of lines to attain a performance increase. For instance, 32-bit PCI requires 49 pins, while 64-bit PCI will presumably need 81 pins [17]. For an IRAM, this bandwidth increase would have the unfortunate side-effect of nearly doubling the number of pins going into the chip.



a) Conventional Architecture



b) IRAM Architecture

Figure 1.Comparison between System-Level Architectures

Fortunately, great advances have recently been made in serial communications technology. Researchers have demonstrated systems that can deliver 2.5 to 4 Gbit/s of I/O [5] [11], and commercial standards including Fibre-Channel Arbitrated Loop (FC-AL) for disks and the upcoming Gigabit Ethernet standard for local-area networks are in place. Current workstations will likely be unable to take advantage of these new devices, due to their I/O system design. This conventional parallel I/O bus design mandates that the fast (e.g., 125 MB/s to 500 MB/s) serial lines be converted from serial to parallel to match the I/O bus, only to be throttled by the bus' low performance. In contrast, IRAMs are a perfect match for such technology.

Figure 1(b) shows how a high-performance serial line can be connected directly to an IRAM chip. Because IRAM has tremendous internal memory bandwidth, it is possible to match virtually any amount of serial I/O bandwidth without interfering with on-chip computation. Single-chip integration provides an additional benefit: by connecting the serial line directly to an IRAM, the chip

1. This research was funded by the Defense Advanced Research Projects Agency (DARPA) F30602-95-C-0014, DARPA DABT63-C-0056, a California State MICRO grant, and an NSF grant CDA-94-01156. Kimberly Keeton is funded by a Lucent Technologies Bell Laboratories Doctoral Fellowship.

architecture can decide on the internal structure necessary to notify the processor of I/O events, keep caches coherent, and update memory.

While a single IRAM with high-bandwidth serial I/O will provide a higher performance and more cost-effective building block for many applications, some applications may require more resources. Several IRAM chips (8 to 16) may also be clustered together on a single SIMM module to form a larger system, which we call SmartSIMM. IRAMs in such a cluster will communicate via high-speed serial links, all connected via a high-speed crossbar switch. SmartSIMMs may also be configured with a smaller number of IRAMs, with high-bandwidth DRAMs (e.g., RAMBUS or synchronous DRAM) filling out the system.

In this paper, we demonstrate that IRAM with serial I/O is an interesting design point for applications. In particular, we focus on non-scientific, commercial workloads. The remainder of the paper is organized as follows. In Section 2, we describe the IRAM and SmartSIMM architectures in greater detail. We examine disk-to-disk sorting in depth in Section 3. Section 4 provides a high-level discussion of on-line transaction processing and decision support workloads, dynamic web content distillation, and high-performance file service. Finally, we present conclusions in Section 5.

2 Proposed Architectures: IRAM and SmartSIMM

2.1 Intelligent RAM (IRAM)

Intelligent RAM, or IRAM, addresses the growing processor-memory performance gap by integrating processing and memory on a single DRAM chip [14] [15]. DRAM is chosen, rather than SRAM, because of its greater memory density and lower cost. IRAM has a number of potential advantages. First, IRAM offers increased (~100X) memory bandwidth and reduced (~10X) memory latency, due to single-chip integration. IRAM also offers reduced (~4X) energy consumption for the memory hierarchy, due to the reduction of off-chip memory accesses through high-capacitance busses [7]. With the addition of a Gbps serial interconnect, IRAM provides significantly higher (10 - 40X) I/O system bandwidth than traditional bus-based systems. IRAM also allows designers to adjust both the size and width of on-chip DRAM.

We propose two prototype IRAMs, IRAM-I and IRAM-II, based on future DRAM generations, as shown in Table 1. A large percentage of the chip is devoted to memory. In addition to the RISC scalar microprocessor, IRAM includes a vector unit. Vector architectures are a natural match for the high bandwidth memory system of an IRAM. In addition, they have been shown to be useful for a much broader class of applications than scientific calculations. Multimedia (video and voice processing), encryption, and some database primitives, such as sorts and hash joins, have been shown to be vectorizable [3] [12] [23]. Vectors present numerous other advantages for IRAM architectures, as well [15].

Characteristic	**IRAM-I (1999)**	**IRAM-II (2002)**
DRAM Generation	256 Mbit	1 Gbit
On-chip Memory (MB)	24	96
Processor Speed (MHz)	300 - 500	500 - 1000
On-chip Memory Latency (ns)	20 - 30	20 - 30
On-chip Memory Bandwidth (GB/s)	50 - 200	50 - 200
I/O Bandwidth via N serial lines (GB/s)	0.5 - 2	0.5 - 4
Vector Performance (GFLOPS)	4	16
Power Consumption (W)	0.5 - 2.0	0.5 - 2.0
Die Size (mm^2)	300	420

TABLE 1. IRAM Target Performance Characteristics

2.2 SmartSIMM

In addition to single-chip IRAM systems, we expect multiple IRAMs to be grouped together on SIMM-like modules into larger systems. The basic architecture of this building block, which we call a SmartSIMM, is shown in Figure 2. A SmartSIMM may contain up to 16 IRAMs, connected via a crossbar switch and Gbps serial lines. In practice, SmartSIMMs may also be configured with a smaller number of IRAMs (i.e., less than four), with high-bandwidth DRAMs completing the system.

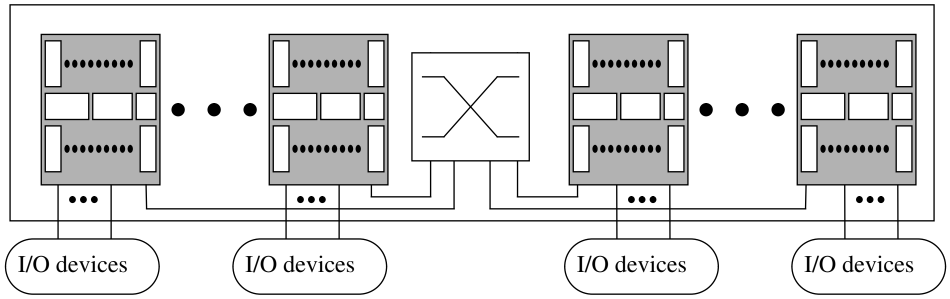


Figure 2.SmartSIMM Architecture

There are many different architectural alternatives for how the resources of this building block will be managed: NUMA, COMA, NOW, etc. All of these (and other) choices are reasonable architectures, where the choice of management influences the application programming model for the SmartSIMM. Given our experience with NOW (network of workstations) environments, we focus on a NOW organization for the application study presented in this paper.

3 Case Study: Disk-to-Disk (External) Sorting

In the next two sections, we demonstrate that IRAM and SmartSIMM will provide an interesting building block for several non-technical applications. Here we present an in-depth discussion of disk-to-disk sorting using MinuteSort, a database industry sorting benchmark. The next section presents a higher-level discussion of IRAM's benefits for several additional commercial and network service applications.

3.1 Disk-to-Disk (External) Sorting

Recognizing that previous sorting benchmarks had become outdated, the authors of AlphaSort introduced MinuteSort to test processor I/O subsystems and operating systems [13]. The benchmark's performance metric is the amount of data (i.e., 100 B records with 10 B keys) that can be sorted from disk to disk in one minute of elapsed time. Price-performance is also computed from the list price of the hardware and the operating system depreciated over three years.

Recently, researchers at Berkeley set the new land-speed record for MinuteSort using a network of UltraSPARC workstations [1]. Their MinuteSort results are shown in Table 2.

System	Processors	Total Disks	Total Memory (GB)	MinuteSort Amount (GB)	MinuteSort Price/Perf. ($/GB)
NOW UltraSPARC (1996)	8	32	1.00	1.4	0.13
NOW UltraSPARC (1996)	16	32	1.0	1.6	0.19
NOW UltraSPARC (1996)	32	64	2.0	3.0	0.20
NOW UltraSPARC (1996)	64	128	4.0	6.0	0.20

TABLE 2. NOWSort MinuteSort Results [1]

Unlike previous sorting champions, whose results were obtained on shared-memory machines with large amounts of memory, their results were obtained on a cluster of workstations with relatively little memory (64 or 128 MB) per node. Throughout their experiments, the authors found that the I/O bus structure of the UltraSparc I workstation provides inadequate support for the disk and network communication needs of high-performance sorting. They were able to achieve reasonable I/O performance only by writing their own software disk striping library. (This is consistent with previous sorting work done in the literature [13].)

Following the discussion in [1], we first present the algorithm for a single node two-pass sort, and then extend the algorithm to the parallel case. For more details on the single-node and parallel two-pass sorts, and their one-pass counterparts, we refer the interested reader to [1].

3.2 IRAM Single Node MinuteSort

NOWSort illustrates a common technique of trading memory for disk accesses [8]. If a large amount of memory is available, we can perform an in-memory, or *one-pass*, sort, where all of the records are read into memory, sorted, and written out to disk. This requires that memory is large enough to hold all of the records to be sorted. If there is insufficient memory to perform a one-pass sort, we can instead execute a *two-pass* sort. The algorithm for a single-node two-pass sort is shown in Figure 3.

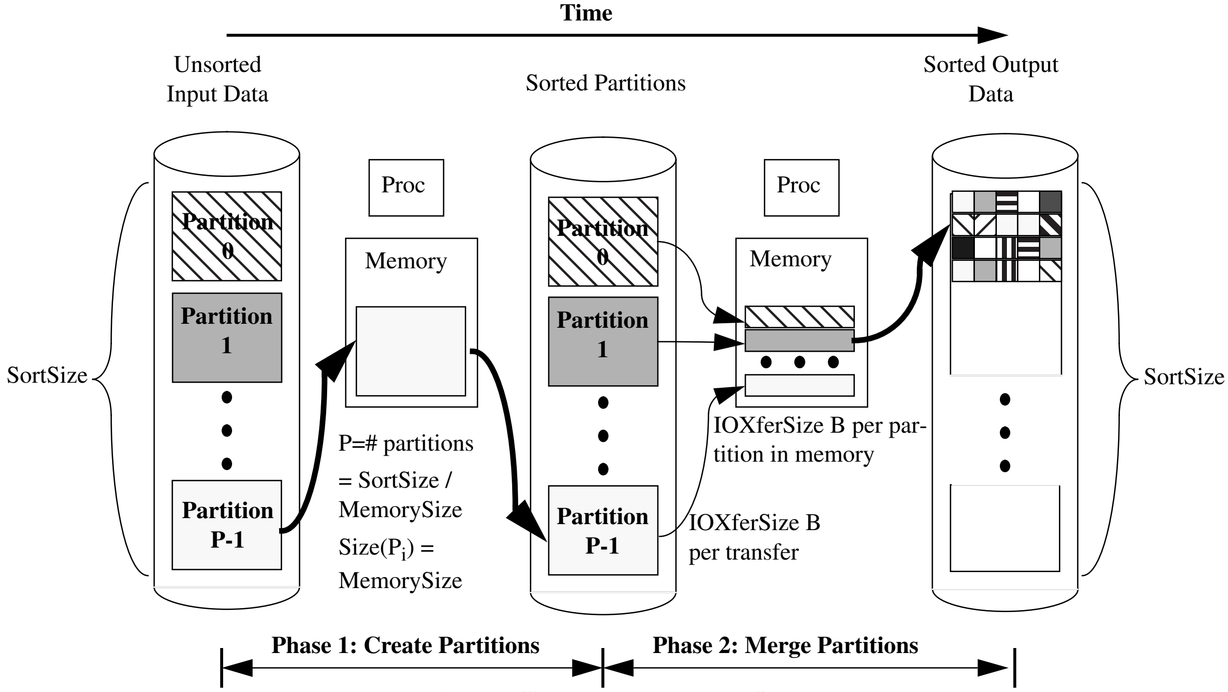


Figure 3.Single-Node Two-Pass Sorting.

During the first phase, *Create Partitions*, a one-pass (in-memory) sort is used multiple times to create `P` sorted partitions (i.e., or runs, as they are commonly referred to in the database literature) on disk. In the subsequent *Merge Partitions* phase, a portion of data (`IOXferSize` B) from each partition is read into memory, merged with the other partitions, and written to disk. This step is repeated until all data has been sorted and written to disk. Performance optimizations for either phase may allocate a fraction of memory to prefetching and/or writing already sorted data. (This optimization is not illustrated in the figure.)

The parallel version of the algorithm uses the same two phases. In the *Create Partitions* phase, each processor reads an entire partition from local disk and separates this data into local and remote buffers. The remote buffers are sent over the network to the appropriate remote processor, and the remainder of local data is received from the network. The processor then sorts its local keys and writes the sorted records to local disk. The parallel *Merge Partitions* phase is entirely local, and identical to the single-node case described above. The interested reader is referred to [1] for more information on these algorithms and their one-pass counterparts.

During the Create Partition phase, the size of each partition is constrained by the size of memory, `MemorySize`. The number of partitions, `P`, can be calculated to be `P = SortSize/MemorySize`. During the second phase, the number of partitions is constrained because a portion of each partition must simultaneously fit into memory: `P*IOXferSize = MemorySize`. Substituting the former equation into the latter, we see that given `MemorySize` GB of memory, we can sort `SortSize` GB of data in a two-pass sort: $\mathtt{SortSize}=\mathtt{MemorySize}^2/\mathtt{IOXferSize}$.

To determine the appropriate single-node IRAM configuration (i.e., how many serial lines and disks per IRAM), we derive some simple relationships between the amount sorted, and the required I/O and memory bandwidths. To sort `SortSize` GB using a two-pass sort, we read and write the entire set of records two times. Thus, MinuteSort requires `4*SortSize/(60 s)` GB/s of average disk I/O bandwidth. In addition, for every `D` GB/s of disk read bandwidth, we require `3*D` GB/s of memory bandwidth [2].

Using these equations, with the constraints on IRAM's IO system bandwidth (e.g., up to 2 GB/s for IRAM-I, up to 4 GB/s for IRAM-II), we can determine the amount that a single IRAM can sort in a minute. For these calculations, we assume that disk characteristics (transfer rates, seek times, rotational latency, density, etc.) will continue to increase (or decrease) at their current rates [9] [10]. These rates, as well as our projected 1999 and 2002 disk characteristics and serial line assumptions are shown below in Table 3 in Appendix A.

For IRAM-I, the largest amount we can sort in a minute is 9 GB, which requires 46 disks and six 2 Gbps serial lines. In this case, the I/O transfer size during the Merge Partitions phase is 64 KB. For IRAM-II, we can sort 36 GB in a minute. This performance uses a 512 KB I/O transfer size during the Merge Partitions phase, and requires 39 disks and seven 4 Gbps serial lines. Even higher single-node performance may be attainable, due to the fact that sorting is vectorizable [23].

We note that the number of required disks (e.g., 46 for IRAM-I and 39 for IRAM-II) is far more disks than needed to store the input data. This seeming inconsistency is due to the limited transfer rate of each disk. In practice, we can utilize only a fraction of the disk's capacity, since only a fraction of the capacity can be transferred in the required time constraint. This "usable" capacity is a small fraction of the available disk capacity: ~440 MB/5.5 GB for IRAM-I, and ~1.2 GB/22.5 GB for IRAM-II.

3.3 Parallel Sort using SmartSIMM

We now turn to the parallel two-pass sort. Recall that in the parallel version, the processor performs network I/O in addition to the local disk I/O, which is the same as that performed in the single-node case. For every `D` GB/s of data read from disk, `2*D` GB/s of network I/O is performed (to send and receive network data). In addition, `9*D` GB/s of memory bandwidth is required [2].

Using the equations, relationships and technology parameters described above, we calculated the MinuteSort amounts for SmartSIMM-I and SmartSIMM-II. These results are shown in detail in Table 4 in Appendix A. SmartSIMM-I, the SIMM module holding 16 IRAM-I chips, can sort at most 72 GB in a minute, using 17 disks and five 2 Gbps serial lines per IRAM. SmartSIMM-II, the IRAM-II system, can sort 288 GB in a minute, utilizing 18 disks and eight 4 Gbps serial lines per IRAM. These systems use 128 KB and 512 KB I/O transfer sizes, respectively. We note that the number of serial lines is the limiting factor in both cases, but especially in the SmartSIMM-II system. Higher sorting performance would be possible if the number of serial lines were greater, or if the bandwidth of each serial line were higher.

3.4 SmartSIMM Cost-Performance

In this section, we examine the cost-performance of the SmartSIMM systems, and compare this utility with that of a comparable NOW. We note that our calculations are only approximate, and should be viewed as such. Our SmartSIMM cost assumptions are based on 1997 retail prices [4] [16], and well-understood technology trends [9] [10], and are shown in Table 5 in Appendix A. We estimate that the cost of an IRAM will be between double and triple the cost of the DRAM of that generation. From existing prices and cost decrease trends, we find that IRAM-I will cost between $405 and $610, and that IRAM-II will cost between $715 and $1070. SmartSIMM prices are computed by summing the cost of the IRAMs, the SIMM modules, the SIMM crossbar interconnect, the disks, and the disk enclosures. These prices are between $239,000 and $242,000 for SmartSIMM-I, and between $224,000 and $230,000 for SmartSIMM-II. We note that these prices are dominated by the disk subsystems: disks comprise 96 to 97% of SmartSIMM-I's total cost, and 93 to 95% of SmartSIMM-II's total cost. Thus, even if the actual cost of IRAM is higher than calculated here, we should not see a drastic change in the overall system price.

For the predicted NOW performance, we have held the 1996 price constant, and extrapolated the performance of 8-, 16-, 32-, and 64-processor NOWs from [1] based on technology trends. Our optimistic estimate is that performance will scale with processor performance, at a rate of 60% per year. Our pessimistic estimate is that performance will scale with disk transfer rates, at 40% per year. Comparisons of the overall system cost vs. performance for SmartSIMMs and NOWs are shown in Figure 4.

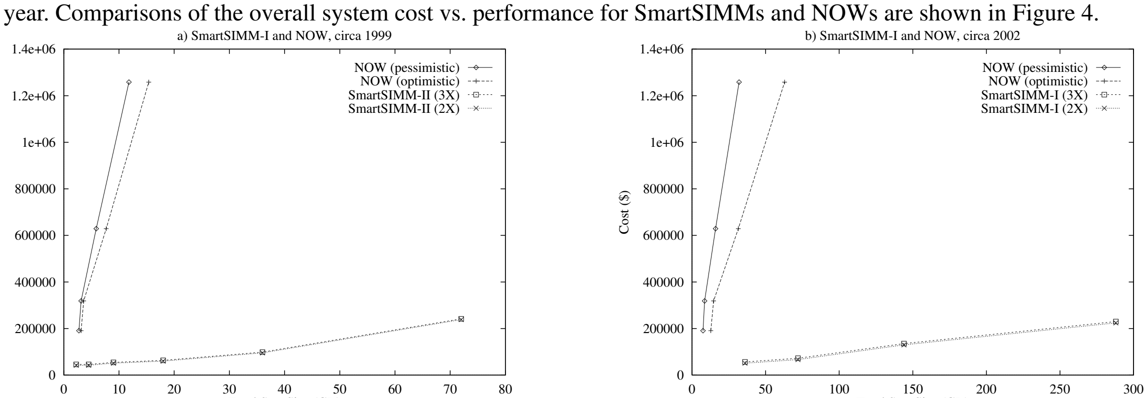


Figure 4.Cost Performance for SmartSIMMs and NOWs

We see that in general, SmartSIMM has significantly better cost-performance than a contemporary NOW. In both the 1999 and 2002 cases, SmartSIMM obtains five times the performance of the optimistic NOW of the same generation. (SmartSIMM achieves performance improvements of an order of magnitude or more, relative to the pessimistic NOW.) This greater SmartSIMM performance is achieved at a fraction of the NOW cost: roughly 20% for both SmartSIMM-I and SmartSIMM-II.

4 Future Work

IRAMs with fast serial I/O lines, and SmartSIMMs composed of these IRAMs can provide the right balance of processing power, memory system performance, and I/O system performance for a wide class of applications. In this section, we briefly discuss some of our ongoing examinations of applications in the context of IRAM and SmartSIMM.

4.1 Decision Support Databases

Decision support (DSS) database workloads, such as TPC-D [21], perform read-only or read-mostly queries over large datasets to answer "what-if" questions.[2] They typically perform large sequential scans over the datasets, and do significant computation on the majority of the data read, to apply the filters comprising the query. Contrary to intuition, decision support workloads are often *compute-bound*, rather than I/O-bound [18]. However, the types of operations performed may benefit greatly from IRAM's vector unit. Decision support workloads join together tables, often using hash-join algorithms. Results are often sorted for in-order or "grouped by" presentation to the user. As described earlier, both hash joins and sorting benefit from vector assist [12] [23]. This computational benefit, in concert with the high I/O and memory bandwidth given by IRAM, will make IRAM and SmartSIMM architectures attractive for this database application.

4.2 Database Transaction Processing

Transaction processing (OLTP) workloads, such as TPC-C [20], have different execution characteristics from their DSS cousins [18]. They perform smaller (i.e., page-sized), more random I/Os. For these data pages, only a small portion of the page is actually utilized in the computation at hand. Sorts are performed less frequently in this workload, but hash joins still commonly used. There are also a great deal of "pointer-chasing" operations, such as manipulating index structures, like B-trees. In general, latency (and in particular, I/O latency) is more critical for OLTP workloads, because a user's transaction must be satisfied within a certain response time constraint. Thus, OLTP databases often employ intelligent buffering and prefetching schemes, and aggressively overlap computation with I/O latency by rescheduling database threads appropriately. Fortunately, there is temporal locality in buffer accesses; as a result OLTP databases typically use large amounts of memory to avoid doing I/O. A rule of thumb recently suggested is that the optimal memory buffer size for a shared memory database running OLTP workloads (i.e., TPC-C in this case) is 10 to 15% of the database size [22]. While this percentage sounds small, this optimal memory size can be greater than 4 GB for larger databases.

The challenge for an IRAM system is to determine how to provide a sufficient amount of memory for buffer caching, while still providing sufficient I/O bandwidth and processing power. A natural choice is a collection of SmartSIMMs, where each SIMM contains a smaller number of processors (e.g., four or less), with high-bandwidth DRAM filling out the remainder of the slots on the SIMM. The high-bandwidth DRAMs could be treated as shared memory, so that a shared memory database could be used, while local IRAM memory could be used as an enormous hardware cache. The trade-off between IRAMs and DRAMs is a topic of our ongoing research.

4.3 Dynamic Content Distillation

An emerging trend in web and Internet service provision is the use of proxy servers to perform on-demand translation between different data formats (e.g., between different image formats such as GIF and JPEG) [6]. Such a service entails high computational requirements to perform this on-demand "dynamic distillation," high network I/O to accept and transmit various document representations, and possibly moderate disk I/O, if the proxy service is also serving as a web cache. IRAM provides an excellent combination of resources for this task: high I/O and memory bandwidth, to easily push and pull the dynamic content, a vector unit that can perform many of the relevant translation algorithms well [3], and the chance to operate several such nodes of a server together on a SIMM, to deal effectively with varying load requirements and fault tolerance.

2. For instance, a business environment DSS query might ask, "how would profits change if we raised the price of product X by 10%?"

4.4 Other Applications

With its high vector performance and high memory bandwidth and capacity, IRAM provides the potential for a new approach to several existing applications [3]. For instance, TCP/IP checksumming, as well as RAID parity calculations, can be vectorized effectively. Data copying, data movement, and memory zeroing (i.e., memcpy or memset) may be performed with higher bandwidth through the vector unit than with traditional scalar processing. IRAM with serial I/O and SmartSIMM will allow us to put a new spin on a large class of existing applications.

5 Conclusions

The performance of current workstations is severely limited by poor I/O bus bandwidth. Intelligent RAM (IRAM) with several Gbps serial lines provides a significantly higher-performance alternative to these conventional systems. The high memory bandwidth provided by IRAM's single integrated processor and memory system, coupled with IRAM's vector unit, and high-speed I/O interfaces will allow us to take a fresh look at existing applications, and to provide a high-performance solution to new problems.

In this paper, we introduced the SmartSIMM, a collection of up to 16 IRAMs on a SIMM module. This larger system may be programmed using any of several different models; these configurations will likely provide higher memory system and I/O performance than is currently available.

We have presented a case study of MinuteSort, a database industry standard sorting benchmark, to illustrate the utility of IRAM and SmartSIMM. We find that SmartSIMM-I, composed of 16 first-generation IRAMs with eight 2 Gbps serial interfaces and 24 MB of on-chip memory each, is capable of sorting 72 GB of data in a minute. The second-generation SmartSIMM-II, with 96 MB of memory and eight 4 Gbps serial interfaces, can sort 288 GB in a minute. This system, however, is somewhat I/O-challenged; SmartSIMM-II would benefit from the several additional serial interfaces. These systems provide an order of magnitude higher performance than contemporary NOW systems, at a small fraction of the cost. We also provided a qualitative discussion of other applications where IRAM and SmartSIMM will provide an interesting architectural building block.

While our performance and cost estimates are only approximate, it is clear that IRAM presents the opportunity to achieve up to an order of magnitude greater performance at lower cost than other alternatives, such as a NOW. However, this additional performance will come only if the I/O system continues to scale with the processor and memory system.

6 References

[1] A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, D. E. Culler, J. M. Hellerstein, and D. A. Patterson. "High-Performance Sorting on Networks of Workstations," to appear in *Proceedings of the 1997 SIGMOD Conference*, May 1997. `http://now.cs.berkeley.edu.`

[2] R. Arpaci-Dusseau. "Empirical Evaluation of Networks of Workstations," Dissertation qualifying exam proposal, April 1997.

[3] K. Asanovic. "Vector Microprocessors," PhD Dissertation, University of California, Berkeley, CA, 1997.

[4] CompUSA Online. `http://www.compusa.com/prd.i/pgen/CompUSA/0L/compusa_menu.htm`.

[5] W. J. Dally and J. Poulton. "Equalized 4Gb/s Signalling", *Hot Interconnects IV Symposium Record*, September 1996.

[6] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. "Adapting to Network and Client Variability via On-Demand Dynamic Distillation," *Proceedings of ASPLOS VII*, October 1996.

[7] R. Fromm, S. Perissakis, N. Cardwell, C. Kozyrakis, B. McGaughy, D. Patterson, T. Anderson, K. Yelick, "The Energy Efficiency of IRAM Architectures," to appear in *Proceedings of ISCA 1997*, June 1997.

[8] J. Gray and G. Graefe. "Storage Metrics," submitted for publication, April 1997.

[9] E. Growchowski. "IBM Leadership in Disk Storage Technology," `http://www.storage.ibm.com.:80/storage/technolo/grochows/grocho01.htm`.

[10] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*, second edition, Morgan Kaufmann, San Mateo, California, 1996.

[11] C. K. Yang and M. A. Horowitz. "A 0.8mm CMOS 2.5 Gb/s Oversampled Receiver for Serial Links," *1996 IEEE ISSCC Digest of Technical Papers*, February 1996.

[12] R. Martin. "A Vectorized Hash Join," unpublished course report, University of California at Berkeley, May 1996, `http://HTTP.CS.Berkeley.EDU/~rmartin/`.

[13] C. Nyberg, T. Barclay, Z. Cvetanovic, J. Gray and D. Lomet. "AlphaSort: A RISC Machine Sort, *Proceedings of the 1994 ACM SIGMOD Conference*, May 1994.

[14] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick. "A Case for Intelligent RAM," *IEEE Micro*, Vol. 17, No. 2, March/April 1997.

[15] S. Perissakis, C. Kozyrakis, T. Anderson, K. Asanovic, N. Cardwell, R. Fromm, K. Keeton, D. Patterson, R. Thomas, and K. Yelick. "Scaling Processors to 1 Billion Transistors and Beyond: IRAM," submitted for publication, March 1997.

[16] Seagate Barracuda 4LP Data Sheet (2.1 GB, 3.5"), Seagate Corporation, `http://www.seagate.com`.

[17] T. Shanley and D. Anderson. *PCI System Architecture*, Third Edition, MindShare, Inc. and Addison-Wesley, 1995.

[18] J. Stephens and R. Raphael, Informix Software, personal communication, April 1997.

[19] N. Talagala, S. Asami, T. Anderson, and D. Patterson. "Tertiary Disk: Large Scale Distributed Storage," submitted for publication, 1997.

[20] Transaction Processing Council (TPC). "TPC Benchmark C Specification," `http://www.tpc.org`.

[21] Transaction Processing Council (TPC). "TPC Benchmark D Specification," `http://www.tpc.org`.

[22] T. Tsuei, A. N. Packer, and K. Ko. "Database Buffer Size Investigation for OLTP Workloads," to appear in *Proceedings of the 1997 SIGMOD Conference*, May 1997.

[23] M. Zagha and G. Blelloch. "Radix Sort for Vector Multiprocessors, "*Proceedings of Supercomputing '91*, November 1991.

A Appendix: Supporting Calculations for SmartSIMM MinuteSort

	Disk Capacity (GB)	Media Transfer Rate (MB/s)	Track Buffer Transfer Rate (MB/s)	Access Time (min. seek + rot. lat.) (ms)	Serial Line Speed (Gbps)
Annual Relationship	1.6	1.4	1.4?	0.7	?
1997 [16]	2.15	15	100	5.07	1
1999	5.50	29	196	2.48	2
2002	22.54	80	540	0.85	4

TABLE 3. Projected I/O Technology Parameters [9] [10]

Characteristic	SmartSIMM-I (1999)	SmartSIMM-II (2002)
Total GB Sorted	72	288
Per-Node Memory Size (MB)	24	96
Per-Node Number of Disks	17	18
Per-Node Number of Serial Lines	5	8
Per-Node Disk Bandwidth (GB/s)	0.3	1.2
Per-Node Network Bandwidth (GB/s)	0.6	2.4
Overall Per-Node I/O Bandwidth (GB/s)	0.9	3.6
Per-Node Memory Bandwidth (GB/s)	2.7	10.8
Total System Cost (IRAM = 2X DRAM) ($)	$239,000	$224,000
Total System Cost (IRAM = 3X DRAM) ($)	$242,000	$230,000

TABLE 4. Summary of SmartSIMM MinuteSort Performance and Price Results

Quantity	1997 Retail Price	IRAM-I/ SmartSIMM-I (256 Mbit DRAM)	IRAM-II/ SmartSIMM-II (1 Gbit DRAM)
Memory Cost / MB (from SIMM) ($)	$11	$6	$3
DRAM Chip Cost ($) [4]	$88	$200	$360
Estimated Minimum IRAM Cost (2X DRAM)	$176	$405	$715
Estimated Maximum IRAM Cost (3X DRAM)	$264	$610	$1070
SmartSIMM Crossbar	N/A	$80	$60
Disk Cost / MB ($)	$0.27	$0.10	$0.02
Total Disk Cost ($) [4]	$600	$550	$490
8-Disk Enclosure [19]	$1900	$1700	$1500

TABLE 5. Estimated Costs for SmartSIMM System Components