Evaluation of Active Disks for Large Decision Support Databases

Mustafa Uysal†, Anurag Acharya‡, Joel Saltz†

Technical Report TRCS99-25
Dept of Computer Science
University of California, Santa Barbara

July 1999

Abstract

Growth and usage trends for large decision support databases indicate that there is a need for architectures that scale the processing power as the dataset grows. To meet this need, several researchers have recently proposed Active Disk architectures which integrate substantial processing power and memory into disk units. In this paper, we evaluate Active Disks for decision support databases. First, we compare the performance of Active Disks with that of existing scalable server architectures: SMP-based conventional disk farms and commodity clusters of PCs. Second, we evaluate the impact of several design choices on the performance of Active Disks. We focus on the performance impact of interconnect bandwidth, amount of disk memory, disk-to-disk communication architecture and the host processor speed on decision support workloads. Our results show that for identical disks, number of processors and I/O interconnect, Active Disks provide better price/performance than both SMP-based conventional disk farms and commodity clusters. Experiments evaluating the impact of design alternatives in Active Disk architectures indicate that: (1) for configurations up to 64 disks, a dual fibre channel arbitrated loop interconnect is sufficient even for the most communication-intensive decision support tasks; (2) most decision support tasks do not require a large amount of memory; (3) direct disk-to-disk communication is necessary for achieving good performance on tasks that repartition all (or a large fraction of) their dataset; and (4) the host processor is rarely a bottleneck for configurations up to 128 disks.

1 Introduction

Growth and usage trends for large decision support databases indicate that there is a need for architectures that scale the processing power as the dataset grows. Results from the 1997 and 1998 Winter Very Large Database surveys document the growth of decision support databases [40, 41]. For example, the Sears Roebuck and Co decision support database grew from 1.3 TB in 1997 to 4.6 TB in 1998. The usage trends indicate that there is a change in user expectations regarding large databases – from primarily archival storage to frequent re-processing in their entirety. Patterson et al [28] quote an observation by Greg Papadopolous - while processors are doubling performance every 18 months, customers are doubling data storage every nine-to-twelve months and would like to "mine" this data overnight to shape their business practices [27].

To meet this need, several researchers have recently proposed Active Disk/IDISK architectures which integrate substantial processing power and memory into disk units [3, 17, 22, 31]. These architectures allow application-specific code to be downloaded and executed on the data that is being read from (written to) disk. Active Disks present a promising architectural direction for two reasons. First, since the number of

0† Dept of Computer Science, University of Maryland, College Park
0‡ Dept of Computer Science, University of California, Santa Barbara

processors scales with the number of disks, Active Disk architectures are well equipped to keep up with the processing requirements for rapidly growing datasets. Second, since the processing components are integrated with the drives, the processing power will evolve as the disk drives evolve. This is similar to the evolution of disk caches – as the drives get faster, the disk caches become larger.

Active Disks can also provide a price-performance advantage over existing server architectures since they offload computation from expensive and powerful central processors to cheap and less-powerful embedded disk processors. Furthermore, Active Disks can offer more cost-effective packaging by sharing cabinetry, cooling fans and power supplies between disks and the embedded processors.

In this paper, we evaluate Active Disk architectures for decision support databases. First, we compare the performance of Active Disks with that of existing scalable server architectures for eight common decision support tasks. In addition to the four algorithms (select, group-by, datacube and sort) we presented in [3], we present four more Active Disk algorithms for decision support tasks: aggregation, project-join queries, maintaining materialized views and determining frequent itemsets for mining association rules from retail transaction data. Initial research on Active Disks has compared the performance of a moderate size Active Disk farm ($\leq$ 32 disks) with that of uniprocessor servers with conventional disks [3, 31]. In this paper, we focus on comparing Active Disks with scalable multiprocessor servers. We consider two alternatives: scalable shared memory machines with large disk farms, and scalable clusters of commodity PCs. We compare performance of these architectures for configurations with the same number of processors and the same number of disks. We evaluate the scalability of these architectures for decision support tasks for configurations with 16-128 disks.

Second, we evaluate the impact of individually scaling several Active Disk components. We focus on the impact of variation in four components: (1) the bandwidth of the I/O interconnect that each Active Disk is attached to, (2) the amount of memory available in an Active Disk, (3) the speed of the front-end processor that manages the Active Disk farm, and (4) the Active Disk communication architecture – i.e., whether direct disk-to-disk communication is permitted. To put these results in perspective, we compare the results of these experiments with those of selected experiments that vary comparable components in SMP/cluster architectures.

Experiments comparing the performance of Active Disks with that of scalable server architectures showed that with identical disks and number of processors, Active Disks provide better price/performance than both SMP-based disk farms and commodity clusters. Active Disks outperform SMP-based disk farms by up to an order of magnitude at a price that is more than an order of magnitude smaller. Active Disks match (and in some cases improve upon) the performance of commodity clusters for less than half the price.

Experiments evaluating the impact of design alternatives in Active Disk architectures indicated that: (1) for configurations up to 64 disks, a dual fibre channel arbitrated loop interconnect is sufficient even for the most communication-intensive decision support tasks; (2) most decision support tasks do not require a large amount of memory; (3) direct disk-to-disk communication is necessary for achieving good performance on tasks that repartition all (or a large fraction of) their dataset; (4) most decision support tasks can be suitably partitioned such that almost all of the processing can be done by disk processors. As a result, the host processor is rarely a bottleneck for configurations up to 128 disks.

2 Evaluation Framework

This research had two objectives. First, to evaluate the performance of Active Disk architectures for large decision support databases and to compare their performance with that of scalable shared memory machines with large disk farms, and scalable clusters of commodity PCs. Second, to evaluate the impact of varying individual Active Disk components. Comparing different architectures is difficult since architectures differ not only in the hardware components but also in the software infrastructure and its ability to utilize the hardware. Furthermore, for a fair comparison, it is important to separately tune each application in the workload for each architecture.

For our experiments, we configured each architecture with identical state-of-the-art disks and scaled each architecture in a similar way – by adding disk-processor pairs. We used four configurations with 16, 32, 64 and 128 disks (and processors). We selected the other components for each architecture based on an existing machine and configuration guidelines suggested by experts. We carefully configured the software components for each architecture so as to optimize the communication and I/O performance (see Section 4). We selected a suite of eight common decision support tasks as the workload. These tasks vary in characteristics such as the amount of computation per byte of I/O, the number of times the entire dataset is read, whether intermediate and final results are written to disk and whether a disk-to-disk shuffle of the input dataset (or a part thereof) is performed. We believe that, taken as a group, these tasks are representative of decision support tasks. For each task, we started with a well-known efficient algorithm from the literature and adapted it for each architecture and the corresponding programming model (see Section 4). Finally, to provide a context within which to interpret the results, we estimated the cost of each configuration and tracked these costs over a one year period.

We conducted all experiments using a simulator called *Howsim* which simulates all three architectures. *Howsim* contains detailed models for disks, networks and the associated libraries and device drivers; it contains coarse-grain models of processors and I/O interconnects. We implemented three versions of each decision support task (optimized for each of the three architectures) and ran them on a DEC Alpha 2100 4/275 workstation to acquire traces recording I/O operations and processing time for user-level tasks. For algorithms that use the amount of memory available as an explicit parameter (sort/join/datacube/materialized-views), we generated traces for multiple memory sizes – to allow us to simulate architectures with different amounts of memory. We used these traces as the workload for our experiments.

2.1 Configurations

For all configurations, we assumed disks similar to the Seagate 39102 from the Cheetah 9LP disk family [32]. These disks have a spindle speed of 10,025 RPM, a formatted media transfer rate of 14.5-21.3 MB/s, an average seek time of 5.4 ms/6.2 ms (read/write) and a maximum seek time of 12.2 ms/13.2 ms (read/write). They support Ultra2 SCSI and dual-loop Fiber Channel interfaces.

Active Disks: For the Active Disk configurations, we assumed that: (1) a Cyrix 6x86 200MX processor (200 MHz) and 32 MB of SDRAM were integrated in the disk units; (2) all the disks were connected by a dual-loop Fiber-channel interface with an aggregate bandwidth of 200 MB/s (100 MB/s per loop); (3) the disks can directly address and communicate with each other using a SCSI-like interface; and (4) communications with clients are handled by a front-end host with a 450 MHz Pentium II and 1 GB RAM.

To study the impact of variation in individual components, we studied alternative configurations that individually scaled: (1) the aggregate bandwidth of the serial interconnect to 400 MB/s; (2) the speed of the processor in the front-end host to 1 GHz; and (3) the memory integrated into the disk unit to 64 MB and 128 MB. No software changes were associated with scaling the bandwidth of the serial interconnect and speed of the front-end processors. To take advantage of the additional memory available in the 64 MB/disk and 128 MB/disk configurations, the number of OS buffers allocated for inter-device communication in each Active Disk is doubled and quadrupled, respectively. This allowed these configurations to tolerate longer communication and I/O latencies.

Clusters: For the cluster configurations, we assumed that each host contained: (1) a 300 MHz Pentium II, (2) 128 MB of SDRAM, (3) a 133 MB/s PCI bus, and (4) a 100BaseT ethernet NIC. We further assumed that the hosts were connected to 24-port 100BaseT ethernet switches with two gigabit ethernet uplinks similar to the 3Com SuperStack II 3900 [26, 33] – the 16 host configuration being connected to a single switch, larger configurations being connected to an array of switches with the uplinks connecting to a gigabit ethernet switch similar to the 3Com SuperStack II 9300 [33, 35]. The network structure has been provisioned to avoid contention in the network and to scale the bisection bandwidth with size of the cluster. We selected these components based on the design of *Avalon* [10], a large commodity PC cluster at the Los Alamos National Lab. We assumed that these machines ran a standard full-function operating system similar to Solaris. Acharya et al [1] report that, on the average, the kernel on a 128 MB Solaris machine has a memory footprint of 24 MB (including the paging free list but not including the file cache); Arpaci-Dusseau et al [8] report a similar result. Accordingly, we assumed that only 104 MB on these hosts is available to user processes.

Shared memory multiprocessors (SMPs): For the SMP configurations, we followed the guidelines for configuring decision support servers (as quoted in [22]): (1) put as many processors in a box as possible to

Component	8/98	11/98	7/99
Seagate 39102	$670	$540	$470
Cyrix 6x86 200Mhz	$32	$30	$22
32 MB SDRAM	$38	$30	$18
Interconnect (per port)	$60	$60	$60
Premium	$ 150	$150	$150
FC host adaptor	$600	$600	$600
Front-end	$9000	$6000	$4200
Approx total	$70000	$58000	$50000

The cost for the front-end host corresponds to a Dell PowerEdge 4300 in 8/98, a 450 Mhz Pentium II, 1 GB Apache Digital (`www.apache.com`) in 11/98 and a 500 Mhz Pentium III, 1 GB machine Apache Digital machine in 7/99. The cost for the FC host bus adaptor corresponds to an LP3000 from Emulex.

Component	8/98	11/98	7/99
Seagate 39102	$670	$540	$470
Cluster node	$1500	$1300	$1150
Network (per port)	$300	$300	$300
Front-end	$9000	$6000	$4200
Approx total	$167000	$ 143000	$108000

The cost for the front-end host corresponds to a Dell PowerEdge 4300 in 8/98, a 450 Mhz Pentium II, 1 GB Apache Digital (`www.apache.com`) in 11/98 and a 500 Mhz Pentium III, 1 GB Apache Digital (`www.apache.com`) machine in 7/99. The cost for the network is based on the two-level 3Com Superstack configuration. The cost of a recently installed chassis-based solution from Lucent (Cajun P550) is similar.

Table 1: Cost evolution for 64-node Active Disk and commodity cluster configurations over a one year period. Costs of most components were taken from `www.pricewatch.com` and www.streetprices.com. Costs of complete systems were taken from vendor web sites. Since the costs of individual components for Active Disks were for retail sales of these components, these numbers overestimate the cost for mass-produced Active Disk units.

amortize the cost of enclosures and interconnects; (2) put as much memory as possible into the box to avoid going to disk as much as possible; and (3) attach as many disks as needed for capacity and stripe data over multiple disks to quickly load information into memory. We assumed an SMP configuration similar to the *SGI Origin 2000*: (1) two-processor boards (with 250 MHz processors) that directly share 128 MB memory; (2) a low-latency high-bandwidth interconnect between these boards ($1\mu s$ latency and 780 MB/s bandwidth); (3) a high-performance block-transfer engine (521 MB/s sustained bandwidth [23]); (4) a high-bandwidth I/O subsystem (two I/O nodes with a total of 1.4 GB/s bandwidth), similar to XIO, that connects to the network interconnect; and (5) a dual-loop Fiber Channel I/O interconnect (200 MB/s) for all disks. Note that the amount of memory is scaled with the number of processors – a 64-processor configuration having 4 GB and a 128-processor configuration having 8 GB. We assumed that these machines ran a standard full-function operating system like IRIX. To determine the impact of varying the I/O interconnect bandwidth, we studied alternative configurations with a 400 MB/s serial interconnect.

2.2 Cost Considerations

We estimated the price of Active Disk configurations using Seagate 39102 as the disk, Cyrix 6x86 200MHz as the embedded processor, 32 MB SDRAM, a dual-loop Fibre Channel interconnect and a substantial *premium* for being a high-end component. We estimated the price of the cluster configuration using a monitor-less Micron PC ClientPro for individual nodes, Seagate ST39102 as the disks and 3Com SuperStack II 3900 and SuperStack II 9300 as the Fast Ethernet and Gigabit Ethernet switches. Table 1 presents the costs of 64-node Active Disks and commodity clusters and their variation over a one year period. We note that over this period, the price of Active Disk configurations is consistently about half that of commodity cluster configurations. Note that since the costs of individual components for Active Disks were for retail sales of these components, these numbers overestimate the cost for mass-produced units. Private conversations with colleagues in the disk manufacturing industry indicate that the volume manufacturing costs would be "quite a bit" less.

We estimated the price of the SMP configuration using the SGI Origin 2000. The Avalon project at Los Alamos Labs quotes the list price of a 64-processor SGI Origin 2000 with 250MHz processors and 8 GB memory to be about $1.8 million [10]. Estimating the cost of 4 GB of SGI memory to be (a generous) $300,000, we estimate the cost of the 64-processor configuration studied in this paper to be about $1.5 million.

2.3 Simulator

We conducted all experiments using a simulator called *Howsim* which simulates all three architectures. *Howsim* contains detailed models for disks, networks and the associated libraries and device drivers; it contains coarse-grain models of processors and I/O interconnects.

For modeling the behavior of disk drives, controllers and device drivers, *Howsim* uses the *Disksim* simulator developed by Ganger et al [16]. *Disksim* has a detailed disk model that supports zoned disks, spare regions, segmented caches, defect management, prefetch algorithms, bus delays and control overheads. *Disksim* has been validated against several disk drives using the published disk specifications and SCSI logic analyzers [16]. For modeling I/O interconnects, *Howsim* uses a simple queue-based model that has parameters for startup latency, transfer speed and the capacity of the interconnect.

For modeling the behavior of networks, message-passing libraries and global synchronization operations, *Howsim* uses the *Netsim* customizable network simulator developed by Uysal et al [36]. *Netsim* models switched networks and an efficient user-space message-passing and global synchronization library with an MPI-like interface. *Netsim* has been validated using a set of microbenchmarks on an IBM SP-2 and a 10-node Alpha SMP cluster with an ATM switch yielding 2-6% accuracy for most messages.

For modeling the behavior of user processes, *Howsim* uses a trace of processing times and I/O requests. It models variation in processor speed by scaling these processing times. To acquire the traces of processing time for user-level tasks, we implemented each algorithm on a DEC Alpha 2100 4/275 workstation. We ran each algorithm with the same dataset and I/O request sizes as used in our experiments. For algorithms that use the amount of memory available as an explicit parameter (sort/join/datacube/materialized-views), we generated traces for multiple memory sizes – to allow us to simulate architectures with different amounts of memory.

For modeling operating system behavior on hosts, *Howsim* uses parameters that represent the time taken for individual operations of interest: `read/write` system calls, context switch time, the time to queue an I/O request in the device-driver and the time to service an I/O interrupt. We obtained the first two using `lmbench` [25] on a 300MHz Pentium II running Linux (10μs for read/write calls, 103μs for context-switch). We charged a fixed cost of 16μs to queue an I/O request in the device-driver.

For Active Disks, *Howsim* models a preliminary implementation of *DiskOS* which provides support for scheduling disklets as well as for managing memory, I/O and stream communication. It uses a modified version of *DiskSim* that is driven by the DiskOS layer. Disklets are written in C and interact with *Howsim* using a stream-based API [3]. For clusters, *Howsim* uses *Netsim* to model the interconnect and *DiskSim* to model the I/O subsystem. An MPI-like message passing interface is used to drive the network model, providing point-to-point communication and global reduction operations. The disk model is driven by a *raw-disk* access library. For SMPs, *Howsim* models two-processor boards connected by a low latency, high bandwidth interconnect. For communication, it models one-way block-transfers, `shmemget` and `shmemput`, as available on the SGI Origin. For synchronization, *Howsim* provides spin-locks, remote queues [12] and global barriers. We used the at-memory fetch-and-op primitive as provided by SGI Origin for spin-locks (which takes about 3μs [21]). *Howsim* models a high-bandwidth I/O subsystem similar to the XIO subsystem in the Origin 2000. The disk model is driven by a *striping library* written on top of the raw-disk access library.

3 Applications

We used a suite of eight common decision support tasks to define the workload for our experiments. These tasks are: (1) SQL select, (2) SQL aggregate, (3) SQL group-by, (4) external sort, (5) the *datacube* operation [18], (6) SQL join, (7) datamining retail data for association rules [5], and (8) maintaining materialized views. These tasks vary in characteristics such as the amount of computation per byte of I/O, the number of times the entire dataset is read, whether intermediate and final results are written to disk and whether a disk-to-disk shuffle of the input dataset (or a part thereof) is performed. We believe that, taken as a group, these tasks are representative of decision support tasks. For each task, we started with a well-known efficient algorithm from the literature and adapted it for each architecture and the corresponding programming model

In this section, we briefly describe the algorithms we used for project-join queries, maintaining materialized views and determining frequent itemsets (for more information, please see [37]). Descriptions of algorithms for select, group-by, datacube and sort can be found in [3, 37]. The algorithm for SQL aggregation is a simple two-level (local/global) aggregation algorithm. In the next section, we present architecture-specific optimizations for these algorithms as well as the software infrastructure used on each architecture.

Project-join query: our project-join algorithm implements a three-phase sort-merge join. The two phases repartition the two participating relations based on the values of the respective join-attributes and create sorted *runs*.[1] The third phase is local to each node. It maintains a heap for the heads of the runs for each relation and performs the join by picking elements from the two heaps (also eliminating duplicates).

Frequent itemsets for datamining: our algorithm for determining frequent itemsets is based on the *eclat* algorithm proposed by Zaki et al [42]. It is an I/O-efficient algorithm that needs only four passes over the data, independent of the size of the largest frequent itemset. The first two passes are processed similar to first two passes of the *count-distribution* algorithm[2] proposed by Agrawal et al [6]. After the first two passes, it clusters the candidate itemsets into equivalence classes and uses these classes to filter, transpose and repartition the input data sets (this constitutes the third pass). The fourth pass is local to each node and uses the transposed version of the data to handle all itemsets with more than two items.[3]

Maintaining materialized views: we have developed our own parallel and I/O-efficient algorithm for maintaining materialized views. For developing this algorithm, we borrowed the ideas of deferred maintenance and self-maintainable views from Mumick et al [13, 30]. Each relation is partitioned over all disks using its (first) join-attribute as the partitioning attribute. An auxiliary view V_i is created for every base relation R_i by applying the appropriate select and project operations. The auxiliary view corresponding to a relation is partitioned the same way as the relation itself and is kept sorted. Updates to the base relations are allowed to proceed immediately. This allows base relations to be consistent at all times. Updates to auxiliary views are deferred by adding the tuples to (unsorted) deltas. Updates to both auxiliary and primary views are propagated at *refresh* time.

4 Architecture-specific Optimizations

For clusters, we assumed an efficient user-space messaging and synchronization library similar to BSPlib [14] that pins send/receive buffers on every host for every communicating peer. We also assumed the `lio_listio` asynchronous I/O interface. We adapted the tasks to use the MPI asynchronous message-passing operations and global synchronization primitives. Taking advantage of the order-independent nature of the processing required by these tasks, we set up the code running on each node to post up to 16 asynchronous receives for *any* message from *any* peer. We adapted all tasks to use large (256 KB) I/O requests and deep request queues (up to four asynchronous requests) to take full advantage of the aggressive I/O subsystem and to overlap the computation with the I/O as much as possible. Since each host can only address its own disk, we partitioned the input datasets over all hosts.

Programming Active Disks is similar to programming clusters with two additional constraints. First, Active Disks are expected to have a limited amount of memory. Due to economic and form-factor limitations, Active Disks are likely to have at most two DRAM chips [7]. This limitation necessitates careful staging of data with an increased degree of pipelining. Second, Active Disks provide a restricted execution environment to preserve data safety and ensure a small footprint for system software (e.g., the stream-based programming model and the DiskOS proposed in [3]). Disk-resident code (*disklets*) cannot initiate I/O operations, cannot allocate (or free) memory, and is sandboxed within the buffers from its input streams and a scratch space that is allocated when the disklet is initialized. In addition, a disklet is not allowed to change where its input streams come from or where its output streams go to. These limitations necessitate that algorithms be structured as coarse-grain data-flow graphs.

With these limitations in mind, we used several techniques to optimize algorithms for Active Disks. To overcome limited amount of memory, we aggressively pipelined partial results, forwarding data to the destination (front-end, media, or peer disk) as soon as the local memory resources are exhausted in an

[1] Size of a run depends upon the amount of memory available on each node.

[2] Riedel et al [31] report an Active Disk algorithm based on *count-distribution*.

[3] In comparison, count-distribution needs k passes where k is the size of the largest candidate itemset.

Active Disk. We adapted our algorithms to use the additional memory at front-end along with the aggregate memory in Active Disks to hold the partial results. We also structured our algorithms for order-independent processing, allowing them to process intermediate results as the data arrives. Restrictions imposed by the stream-based programming model have turned out to be beneficial for efficiency. We aggressively batched I/O operations at the host. This provides valuable information on the access patterns to the DiskOS and enables important optimizations such as prefetching and overlapping computation, communication, and I/O.

For SMPs, we assumed an implementation of the *remote queue* abstraction (as suggested by Brewer et al [12]), the `lio_listio` asynchronous I/O interface and user-controllable disk striping for individual files. We adapted the tasks to use one-way block-transfers (`shmemput/shmemget`) and remote queues for moving data between processors. Given the volume of data being transferred and the one-way nature of the data movement, block-transfers and remote queues are well-suited for these tasks. We striped each file over all disks using a 64 KB chunk per disk. To take advantage of the aggressive I/O subsystem, each processor issues up to four 256 KB asynchronous requests (each request transferring a 64 KB chunk from each of four disks). However, for `sort` and `join`, which shuffle their entire dataset and write it back to disk, we partitioned the disks into separate read and write groups (as in NOW-sort [8]). Since all processors can address all disks, we did not a-priori partition the input datasets to processors. Instead, we maintained two shared queues (read/write) of fixed-size blocks in the order they appear on disk. When idle, each processor locks the queue and grabs the next block off the queue. This technique reduces the seek costs at the disks as the overall sequence of requests roughly follows the order in which data has been laid out on disk. A-priori partitioning of the dataset would result in a potentially long seek for every request.

5 Experimental Results

For the core set of experiments, we evaluated the performance of all eight decision support tasks on all three architectures for 16-disk, 32-disk, 64-disk and 128-disk configurations. We performed additional experiments to evaluate the impact of scaling various Active Disk components. To put these results in perspective, we also performed selected experiments that varied comparable components in SMP/cluster architectures. We focused on the impact of variation in four components: (1) the bandwidth of the I/O interconnect that each Active Disk is attached to, (2) the amount of memory available in an Active Disk, (3) the speed of the front-end processor that manages the Active Disk farm, and (4) the Active Disk communication architecture – i.e., whether direct disk-to-disk communication is permitted.

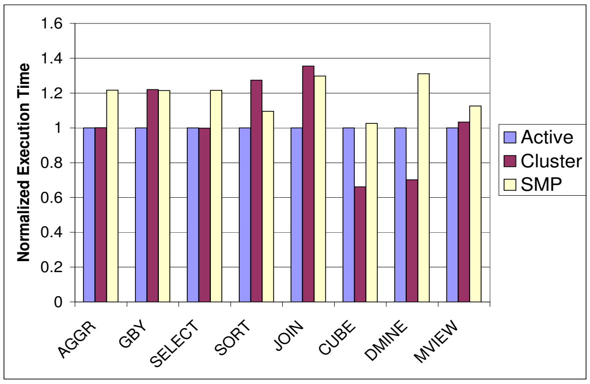
For these experiments, we constructed large datasets for each tasks. The dataset for each task was based on the dataset used in the papers that describe the original algorithms [4, 5, 8]. We used 16 GB datasets for all applications except `join` for which we used a 32 GB dataset and `mview` for which we used a 15 GB dataset. Table 2 presents the salient features of the dataset used for each task.

Task	Characteristics of Dataset
`select`	268 million, 64-byte tuples, 1% selectivity
`aggregate`	268 million, 64-byte tuples, SUM function
`groupby`	268 million, 64-byte tuples, 13.5 million distinct
`dcube`	536 million, 32-byte tuples, 4-dimensions, 1%,0.1%,0.01% and 0.001% distinct values
`sort`	100-byte tuples, 10-byte uniformly distributed keys
`join`	64-byte tuples, 4-byte keys (uniformly distributed), 32-byte tuples after projection
`dmine`	300 million transactions, 1 million items, avg 4 items per transaction, 0.1% minsup
`mview`	32-byte tuples, 4 GB derived relations, 1 GB deltas

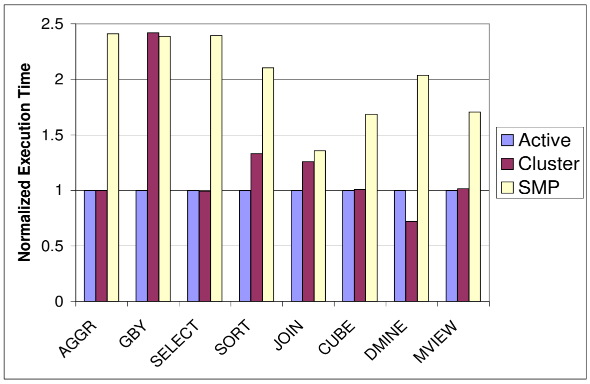
Table 2: Datasets for the tasks in the workload.

5.1 Performance comparison between the three architectures

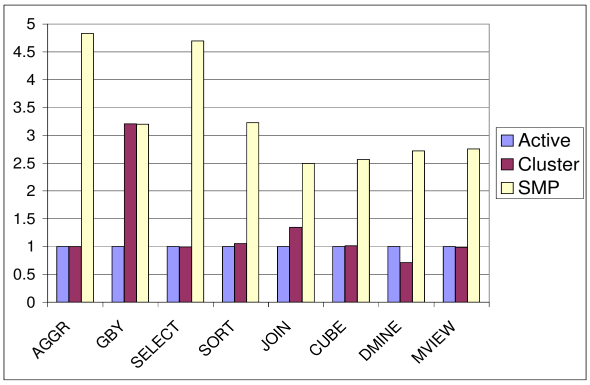
Figure 1 compares the performance of all eight tasks on comparable configurations of all three architectures. The results for each task on configurations of a particular size (16/32/64/128 disks) are normalized with respect to the performance of the same task on the Active Disk configuration of the same size. We make three observations. First, for the 16-disk configurations, the performance of all three architectures is comparable. This shows that the implementations on all three architectures are well-optimized.



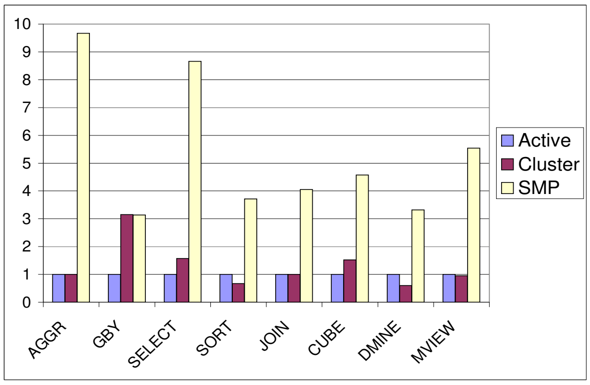
(a) 16 disk configurations



(b) 32 disk configurations



(c) 64 disk configurations



(d) 128 disk configurations

Figure 1: Performance of all eight tasks on comparable configurations of Active Disks, clusters and SMPs. The results for each task on configurations of a particular size (16/32/64/128 disks) are normalized with respect to the performance of the same task on the Active Disk configuration of the same size. Note that the range of the y-axis is *different* for every graph.

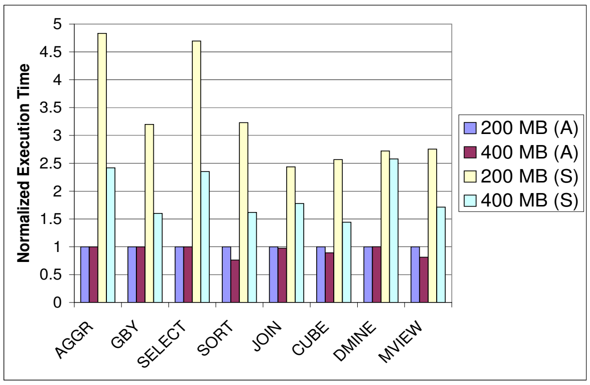
Second, for larger configurations, Active Disks perform significantly better than corresponding SMP configurations; the difference in their performance grows with the size of the configuration. For 32-disk configurations, the tasks in our suite run 1.4-2.4 times faster on Active Disks; for 128-disk configurations, the speedup is 3-9.5 fold. The largest performance differences (8.5-9.5 fold on 128-disk configurations) occur for tasks that allow large data reductions on Active Disks (e.g., `aggregate/select`). However, even tasks that repartition all or part of their input dataset (`sort/join/dmine/mview`) are significantly faster (4-6 fold on 128-disk configurations) on Active Disks. To put these results in context, note that the estimated price of the 64-disk Active Disk configuration is more than an order of magnitude smaller than that of the corresponding SMP configuration.

Third, the performance of Active Disks and cluster configurations is comparable. For all tasks other than `group-by`, the ratio of Active Disk performance to cluster performance is within 0.75 and 1.5 for all configurations. The performance differences between these two architectures arise mainly from application characteristics and the characteristics of the network configurations. Active Disk configurations studied in this paper use a pair of serial Fibre Channel Arbitrated Loops whose effective bisection bandwidth does not grow with the number of disks but which can provide up to 200 MB/s bandwidth into a single node. On the other hand, the cluster configurations use a two-level fat-tree-like network (with Fast Ethernet/Gigabit Ethernet links) whose effective bisection bandwidth grows with the number of hosts but which limits the bandwidth into a single node to 100 Mb/s. The larger per-link bandwidth of the serial interconnect allows Active Disks to perform somewhat better for the smaller configurations whereas the growing bisection bandwidth of the fat-tree-like structure allows cluster configurations to catch up or do better for larger configurations. This is particularly relevant for tasks that repartition all or most of their input data (`sort/join/dmine/mview`). The performance of `group-by` on cluster configurations is limited by end-point congestion at the front-end. This task delivers a large volume of data to the front-end whose 100Mb/s connection becomes a bottleneck. In contrast, the 200 MB/s link provided by Fibre Channel allows Active Disk configurations to scale better. To put these results in context, note that the estimated price of a 64-disk Active Disk configuration is consistently (over a one year period) about half the price of a comparable commodity cluster configuration.

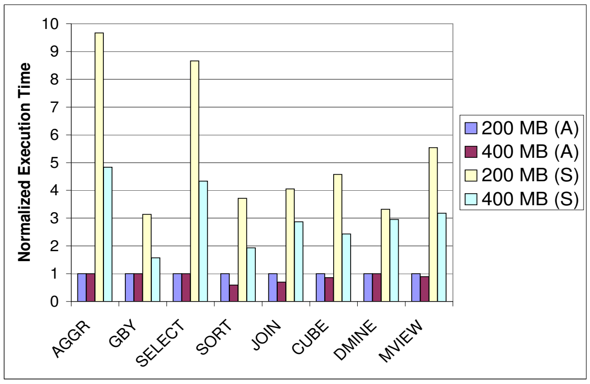
5.2 Impact of variation in interconnect bandwidth

Active Disks appear to achieve their performance advantage over SMPs by reducing the amount of data being sent over the I/O interconnect. For example, the entire dataset for `sort` passes over the I/O interconnect four times for SMP configurations and only once for Active Disk configurations. To study the impact of variation in I/O interconnect bandwidth, we conducted additional experiments with Active Disk and SMP configurations assuming a 400 MB/s serial I/O interconnect. Figure 2 presents the results.

We note that doubling the I/O interconnect bandwidth has a large impact on the performance of SMP configurations for all tasks. This indicates that the I/O interconnect is a major bottleneck for these tasks on SMP configurations. Note, however, that Active Disk configurations with a 200 MB/s I/O interconnect outperform SMP configurations with a 400 MB/s I/O interconnect (1.5-4.8 times faster for these tasks on 128-disk configurations).



(a) 64-disk configurations

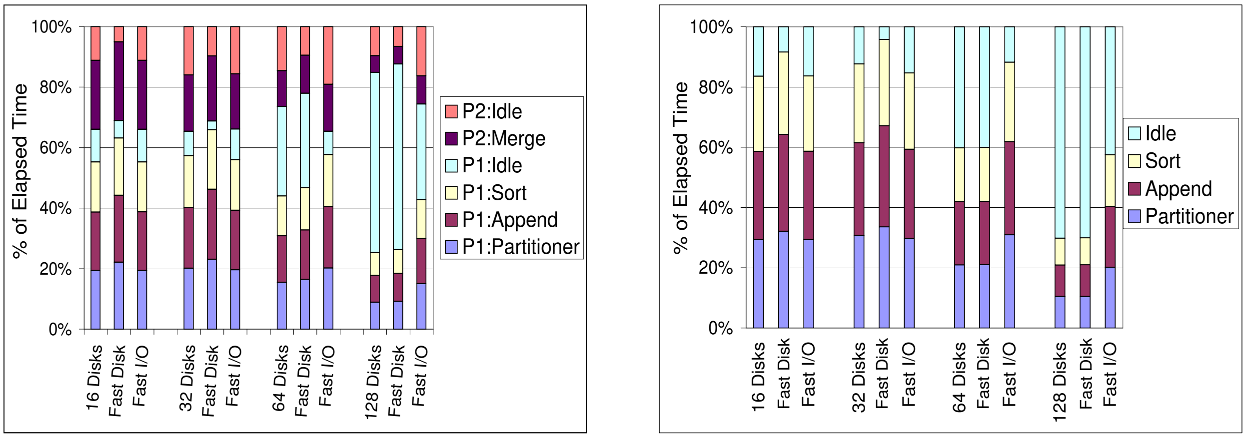


(b) 128-disk configurations

Figure 2: Impact of varying the I/O interconnect bandwidth for Active Disks and SMPs. The results for each task on a given configuration are normalized with respect to the performance of the same task on the Active Disk configuration of the same size. In the legend, **A** stands for Active Disks and **S** stands for SMP.

For the Active Disk configurations, only the tasks that repartition all (or a large fraction) of their dataset, (i.e., `sort`, `join` and `mview`) achieve some, albeit small, performance improvement. Achieving good performance for these tasks can be challenging as they require careful coordination of data movement. To better understand the impact of I/O interconnect bandwidth on the performance of such tasks, we studied `sort` (as a representative task) in more detail. We selected `sort` because it shows the largest improvement with the increase in I/O interconnect bandwidth.

Figure 3 presents a detailed breakdown of the execution time for `sort` on a variety of Active Disk configurations. It presents a breakdown of the execution time as well as the impact of evolution of individual components. The bars labeled *16 Disks*, *32 Disks* and so on correspond to the configurations described in Section 2.1. The bars labeled *Fast Disk* correspond to configurations with upgraded disks. We assumed disks similar to the Hitachi DK3E1T-91 with 12030 RPM, 18.3-27.3 MB/s media bandwidth, 5 ms/6 ms (read/write) average seek time, 10.5 ms/11.5 ms (read/write) max seek time [24]. The bars labeled *Fast I/O* correspond to configurations with a 400 MB/s I/O interconnect.



(a) Both phases.

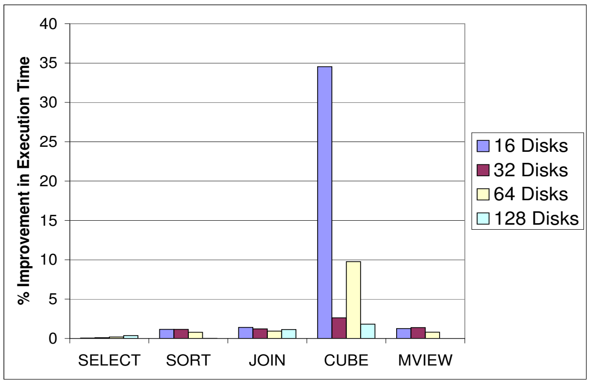
(b) Sort phase

Figure 3: Performance breakdown of `sort` on Active Disk configurations. P1 refers to the first (sort) phase and P2 refers to the second (merge) phase. In the legend, *P1:Partitioner* is the time spent in the `partitioner` disklet partitioning tuples; *P1:Append* is the time spent in the `sorter` disklet collecting data sent by different disks; *P1:Sort* is the time spent in the `sorter` disklet sorting individual runs; and *P1:Idle* is total CPU idle time in the sort phase. This includes waiting for disk media read/writes and network I/O. In the merge phase, *P2:Merge* is the time spent merging the sorted runs in memory and *P2:Idle* is the CPU idle time spent waiting for disk media read/writes.

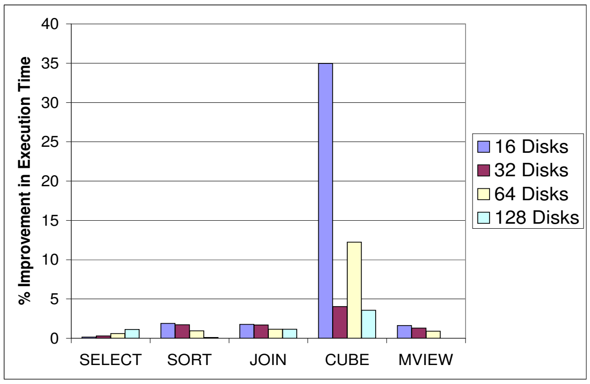
Figure 3 (a) presents the breakdown for the complete execution of `sort`. We note that the sort phase (which repartitions the dataset) dominates the execution time for all configurations. Figure 3 (b) zooms in on the sort phase. We note that, for 16/32/64-disk configurations, the computation is well-balanced and that the total idle time (P1:Idle+P2:Idle) is a small fraction of the total execution time. This indicates that up to 64-disk configurations, neither the I/O interconnect, nor the disk media is a bottleneck. Accordingly, upgrading either the disks or the I/O interconnect makes only a small difference. For the 128-disk configuration, however, idle time dominates the execution time. A comparison with the *Fast-Disk* and *Fast-I/O* bars shows that the upgrading the disks makes little difference whereas upgrading the I/O interconnect has a major impact on performance. From this we conclude that a dual-loop Fibre Channel Arbitrated Loop interconnect can meet the demands of challenging tasks up to 64 disks. To scale to configurations larger than the ones examined in this paper, we recommend a more aggressive interconnect (e.g., multiple Fibre Channel loops connected by a FibreSwitch).

5.3 Impact of variation in disk memory

Economic and form-factor limitations indicate that Active Disks are likely to have at most two DRAM chips [7]. This limitation necessitates careful staging of data with an increased degree of pipelining. To determine the impact of the limited memory capacity of Active Disks on the performance of decision support tasks, we compared the performance of Active Disk configurations with different amounts of memory. Our results indicate that the performance of `aggregate`, `groupby` and `dmine` on Active Disks did not improve with additional memory. Since `aggregate` computes a zero-dimensional aggregate, its performance is naturally insensitive to the amount of memory available. The performance of `groupby` is dominated (for 64 disks and beyond) by the time to transfer results to the front-end host. As a result, adding more memory makes little difference to its performance. The memory usage of `dmine` depends on the number of itemset frequency counters it needs to maintain; this, in turn, depends on its input parameters (number of items and minimum support). For the dataset used in our experiments, the frequency counters needed 5.4 MB per disk. As a result, our experiments show no variation in the performance of `dmine` with increase in memory size. For much larger datasets with orders of magnitude more items and/or much smaller minimum support, the memory size may become relevant.



(a) 64 MB disk memory



(b) 128 MB disk memory

Figure 4: Impact of varying the memory available on Active Disks.

For the remaining tasks, Figure 4 presents the performance improvements obtained by increasing the size of the memory to 64 MB and 128 MB. We note that for tasks other than `dcube`, increasing the memory makes a negligible ($\leq 2\%$) difference in the performance. To better understand this result, we examined the performance of `sort` more closely. Increasing the memory available in Active Disks allows the creation of longer runs. We found that longer runs resulted in small decreases in both processing time and disk access time. Switching from 40 runs of 25 MB each (used for 32 MB Active Disks) to 20 runs of 50 MB each (used for 64 MB Active Disks) reduced the CPU cost by about 7% and the disk access time by about 2%.

The algorithm used in our `dcube` implementation is based on the *PipeHash* algorithm proposed by Agarwal et al [4]. This algorithm tries to minimize the number of passes by scheduling several group-bys as a pipeline. The memory requirement for this task is determined by the sizes of the hash tables needed for the component group-bys. For the 16-disk configuration, the total disk memory is 512 MB. The size of the hash table for the largest group-by is 695 MB. As a result, for this group-by each disk forwards partially computed hash tables to the front-end host (which has 1 GB memory). Increasing the disk memory for the 16-disk configuration provides substantial performance improvement as the disks no longer have to forward intermediate results to the front-end. Note that there is no performance improvement beyond 64 MB since at that point, all group-bys fit in disk memory. Another threshold is reached at 64-disk configurations where increasing the amount of memory allows the algorithm to reduce the number of passes from three to two. The dataset used in our experiments requires computation of 15 group-bys. Of these, 14 group-bys can be merged into a single scan if a total of 2.3 GB is available at the disks. A 64-disk configuration with 32 MB/disk has 2 GB; with 64 MB/disk, it has 4 GB. This is the cause of the spike in performance improvement graphs at 64-disk configurations.

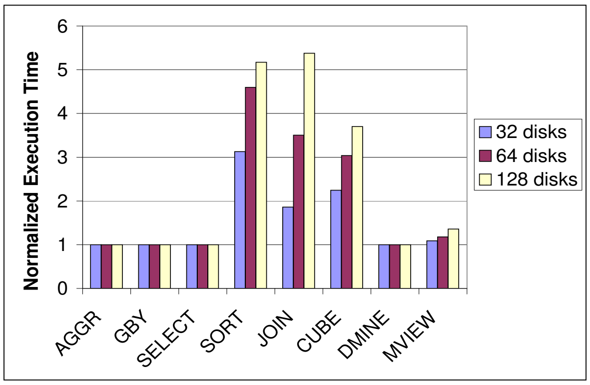
Note that even for `dcube`, the largest performance improvement is only about 35% which occurs for 16-disk configurations; for larger configurations, the performance improvement is less than 12%. To put this result in perspective, note that the performance of `dcube` on SMPs is a factor of 2.5 slower than on Active Disks for 64-disk configurations and a factor of 4.5 slower for 128-disk configurations. On the commodity cluster configurations, the performance of `dcube` is about 35% faster than Active Disks for 16 disks and about 40% *slower* for 128 disks.

5.4 Impact of direct disk-to-disk communication

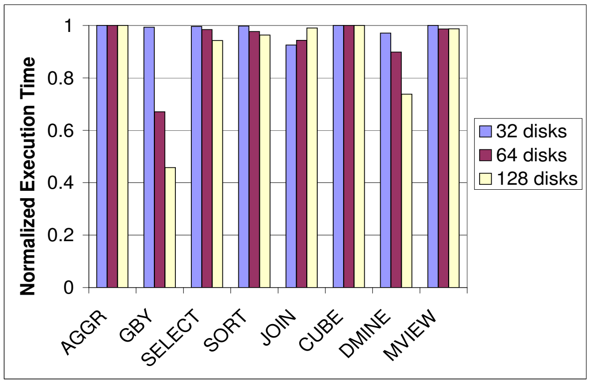
In traditional servers, data transfers between I/O devices occur through the host to which they are connected. Two of the initial Active Disk proposals [3, 31], including our own, assumed a similar architecture. A third proposal [22] proposed direct disk-to-disk communication but did not present experimental evidence for its necessity. In the core set of experiments for this paper, we assumed that disks could communicate directly without routing the data through the host memory. To understand the impact of this decision, we conducted additional experiments that restricted Active Disks to communicate only with the front-end host. Figure 5 (a) presents the results. We note that this restriction has a large impact (up to a five-fold slowdown) for the three communication-intensive tasks and virtually no impact on the remaining six tasks. Although supporting direct disk-to-disk communication complicates the DiskOS and increases its memory footprint, the substantial impact it has on the performance of important applications, leads us to conclude that Active Disk architectures should allow direct disk-to-disk communication.

5.5 Impact of variation in host processor speed

Efficient Active Disk algorithms offload most of the processing from the host. However, coordination and scheduling of disklets, the combination of results from individual disks, and the delivery of final results to clients are still handled by the host. This can result in the host processor becoming a bottleneck for large configurations or for applications that deliver large volumes of data to clients. To determine the impact of host processor speed on decision support tasks, we performed experiments with configurations with a 1 GHz host processor. Figure 5 (b) presents the results. Our results indicate that for almost all tasks, the impact of more than doubling the host processor speed is small even for the largest configurations. This indicates that most decision support tasks can be partitioned such that the host processor is not a bottleneck. For `group-by`, however, increasing the host processor speed has a substantial impact on performance. Furthermore, the performance impact increases roughly linearly with configuration size. As mentioned earlier, `group-by` delivers a large volume of data to the front-end. The only other task that leveraged significant performance improvement (25% for the 128-disk configuration, less than 10% for smaller configurations) from a faster host processor was `dmine`. The first phase of `dmine` computes local itemset frequency counters at each disk and uses the host processor to combine the results.



(a) Restricted communication



(b) 1 GHz host processor

Figure 5: (a) Impact of a restricted communication architecture; (b) impact of variation in host processor speed. The results for each task on a given configuration are normalized with respect to the performance of the same task on the core Active Disk configuration of the same size.

6 Related work

Several researchers have recently explored the concept of programmable disks [3, 11, 15, 19, 22, 31, 38]. Initial studies on using Active Disks for databases compared the performance of a moderate size Active Disk farm ($\leq$ 32 disks) with that of uniprocessor servers with conventional disks [3, 31]. They demonstrated the potential of Active Disk architectures for achieving performance gains for database workloads. In this research, we have examined much larger configurations and have compared the performance of Active Disks with that of scalable multiprocessor servers for a large suite of decision support tasks. We have also examined the impact of several design choices for Active Disk architectures.

Keeton et al [22] and Hsu et al [19] analyzed technological trends and individual applications to evaluate the expected performance of Active Disk architectures for decision support tasks. Keeton et al [22] proposed an architecture (IDISK) in which a processor-in-memory chip (IRAM [28]) is integrated into the disk unit, the disk units are connected by a crossbar and can communicate directly. They argue that IDISK architectures offer several potential price and performance advantages over traditional server architectures for decision support. Our results regarding the impact of direct disk-to-disk communication provide empirical evidence in support of their design decision. Hsu et al [19] evaluated an architecture called *SmartSTOR* which incorporates a high-end processor and one or more commodity disks in a single storage box. Based on analytical projections from actual runs of the TPC-D benchmark, they conclude that in such an architecture, the power of disk processors has a significant impact on the performance.

In the core set of experiments presented in this paper, we compared the performance of Active Disks, commodity clusters and SMPs for decision support tasks. These experiments demonstrated that inexpensive architectures that scale I/O bandwidth with the number of processors (Active Disks and commodity clusters) can significantly outperform expensive machines for decision support tasks. Similar results have been reported by [9, 34]. Arpaci-Dusseau et al [9] compared the performance of a cluster of SUN Ultra 1 workstations and an Ultra Enterprise 5000 SMP for scan, sort, and transpose; Tamura et al [34] have compared the performance of a cluster of 100 PCs connected with an ATM switch for TPC-D and data mining. Both these studies demonstrate that a commodity cluster can outperform configurations based on large machines.

The idea of embedding a programmable processor in a disk is not new. The I/O processors in the IBM 360 allowed users to download *channel programs* that were able to make I/O requests on behalf of the host programs [29]. Database machines in the late 1970s proposed various levels of processor integration into disk: processor per track, processor per head, and "off-the-disk" designs that used a shared disk cache with multiple processors and multiple disks ([20] contains a review of database machines). There are several differences between the database machines of late 1970's and today's Active Disks. Unlike the database machine processors, the processors proposed for Active Disks are general-purpose commodity components and are likely to evolve as the disks evolve. Unlike the low-bandwidth interconnects used in database machines, today's serial I/O interconnects such as Fibre Channel provide high bandwidth for commodity disk drives. Unlike the limited repertoire of operations performed by database machines, Active Disks take advantage of algorithmic research for shared-nothing architectures to provide efficient implementations of a wide variety of operations. Furthermore, Active Disks can be used to perform operations for non-relational data such as image processing [3, 31] and file-system and security-related processing [11, 15, 38, 39].

7 Conclusions

In this paper, we have evaluated Active Disk architectures for decision support databases and have compared their performance, scalability and price with that of existing scalable server architectures. In addition, we have examined the impact of individually scaling several Active Disk components.

Experiments comparing the performance of Active Disks with that of scalable server architectures showed that with identical disks and number of processors, Active Disks provide better price/performance than both SMP-based disk farms and commodity clusters. Active Disks outperform SMP-based disk farms by up to an order of magnitude at a price that is more than an order of magnitude smaller. Active Disks match (and in some cases improve upon) the performance of commodity clusters for less than half the price.

Experiments evaluating the impact of design alternatives in Active Disk architectures indicated that:

- For configurations up to 64 disks, a dual fibre channel arbitrated loop interconnect is sufficient even for the most communication-intensive decision support tasks. To scale to larger configurations, a more aggressive interconnect (e.g., multiple fibre channel loops connected by a FibreSwitch) would be needed.
- Most decision support tasks do not require a large amount of memory. Of the tasks examined in this paper, increasing the disk memory to 64 MB and 128 MB provided a significant performance gain only for the datacube task. Even for datacube, the performance gain is small ($\sim$ 35%) and is mainly achieved on small configurations.
- Direct disk-to-disk communication is necessary for achieving good performance on tasks that repartition all (or a large fraction of) their dataset (i.e., sort, join, materialized views). Requiring disk-to-disk communication to pass through host memory resulted in slowdowns up to a factor of five.
- Most decision support tasks can be suitably partitioned such that almost all of the processing can be done by disk processors. As a result, the host processor is rarely a bottleneck for configurations up to 128 disks. This conclusion, however, does not hold for tasks (such as group-by) that can deliver a very large volume of data to the requesting client.

Finally, we note that it was fairly easy to adapt and optimize existing distributed-memory algorithms for Active Disks. This allowed us to rapidly implement and optimize a large suite of decision support tasks. In fact, it was significantly easier to implement and optimize these algorithms for Active Disks than for clusters. We believe that the coarse-grain dataflow nature of the stream-based programming model used for Active Disks was largely responsible for this. Given the volume of data processed and the cost of fetching data from disk, optimizing I/O-intensive algorithms often is matter of setting up efficient pipelines where each stage performs some processing on the data being read from disk and passes it on to the next stage [2, 8].

References

[1] A. Acharya and S. Setia. Availability and utility of idle memory in workstation clusters. In *Proc. of SIGMETRICS'99*, pages 35–46, 1999.

[2] A. Acharya, M. Uysal, R. Bennett, A. Mendelson, M. Beynon, J. Hollingsworth, J. Saltz, and A. Sussman. Tuning the performance of I/O-intensive parallel applications. In *Proceedings of the Fourth ACM Workshop on I/O in Parallel and Distributed Systems*, May 1996.

[3] A. Acharya, M. Uysal, and J. Saltz. Active Disks: Programming Model, Algorithms and Evaluation. In *Proceedings of ASPLOS VIII*, pages 81–91, October 1998.

[4] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proceedings of the 22nd International Conference on Very Large Databases*, pages 506–21, 1996.

[5] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference on Management of Data*, pages 207–16, 1993.

[6] R. Agrawal and J. Shafer. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–9, 1996.

[7] D. Anderson. Considerations for smarter storage devices. Talk at NASD workshop on storage embedded computing[4], June 1998.

[8] A. Arpaci-Dusseau, R. Arpaci-Dusseau, D. Culler, J. Hellerstein, and D. Patterson. High-performance sorting on networks of workstations. In *Proceedings of SIGMOD'97*, 1997.

[4] *www.nsic.org/nasd/1998-jun/anderson.pdf*

[9] R. Arpaci-Dusseau, A. Arpaci-Dusseau, D. Culler, J. Hellerstein, and D. Patterson. The architectural costs of streaming I/O: A comparison of workstations, clusters, and SMPs. In *Fourth Symposium on High Performance Computer Architecture (HPCA4)*, 1998.

[10] The Avalon Home Page[5], 1998. Avalon is a 140-processor Beowulf cluster at Los Alamos National Lab.

[11] E. Borowsky, R. Golding, A. Merchant, L. Schrier, E. Shriver, M. Spasojevic, and J. Wilkes. Using attribute-managed storage to achieve QoS. In *Proceedings of the 5th International Workshop on Quality of Service*, 1997.

[12] E. Brewer, F. Chong, L. Liu, S. Sharma, and J. Kubiatowicz. Remote queues: Exposing message queues for optimization and atomicity. In *Proc. of the 7th SPAA*, pages 42–53, 1995.

[13] L. Colby, T. Griffin, L. Libkin, I. Mumick, and H. Trickey. Algorithms for deferred view maintenance. In *Proc. of SIGMOD'97*, pages 469–80, 1997.

[14] S. Donaldson, J. Hill, and D. Skillicorn. BSP Clusters: High Performance, Reliable and Very Low Cost. Technical Report PRG-TR-5-98, Oxford University Computing Laboratory, 1998.

[15] G. Gibson et al. File server scaling with network-attached secure disks. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics '97)*, 1997.

[16] G. Ganger, B. Worthington, and Y. Patt. The DiskSim Simulation Environment Version 1.0 Reference Manual[6]. Technical Report CSE-TR-358-98, Dept of Electrical Engineering and Computer Science, Feb 1998.

[17] J. Gray. Put EVERYTHING in the Storage Device. Talk at NASD workshop on storage embedded computing[7], June 1998.

[18] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Proceedings of the 12th International Conference on Data Engineering*, pages 152–9, New Orleans, February 1996.

[19] W. Hsu, A. Smith, and H. Young. Projecting the performance of decision support workloads on systems with smart storage (SmartSTOR). Private Communication, Feb 1999.

[20] A.R. Hurson, L.L. Miller, and S.H. Pakzad, editors. *Parallel Architectures for Database Systems*. IEEE Computer Society Press, 1989.

[21] D. Jiang and J. Singh. A methodology and an evaluation of the SGI Origin 2000. In *Proc. of the Intl. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 171–81, Madison, WI, June 1998.

[22] K. Keeton, D. Patterson, and J. Hellerstein. The Case for Intelligent Disks (IDISKS). *SIGMOD Record*, 27(3), 1998.

[23] J. Laudon and D. Lenoski. The SGI Origin: a ccNUMA highly scalable server. In *In Proc. of Intl. Symposium on Computer Architecture*, pages 241–51, Denver, CO, June 1997.

[24] Hitachi America Ltd. Datasheet for the Hitachi DK3E1T-91[8], 1998.

[25] L. McVoy and C. Staelin. lmbench: portable tools for performance analysis. In *In Proc. of 1996 USENIX Technical Conference*, Jan 1996.

[26] Mier Communications. Evaluation of 10/100 BaseT Switches. *http://www.mier.com/reports/cisco/-cisco2916mxl.pdf*, April 1998.

[5] http://cnls.lanl.gov/avalon/

[6] Available at http://www.ece.cmu.edu/ ganger/disksim/disksim1.0.tar.gz

[7] *http://www.nsic.org/nasd/1998-jun/gray.pdf*

[8] Available at *www.hitachi.com/storage/docs/dk3e1t-91.pdf*

[27] G. Papadopolous. The future of computing. Unpublished talk at NOW Workshop, July 1997.

[28] D. Patterson et al. Intelligent RAM (IRAM): the Industrial Setting, Applications, and Architectures. In *Proceedings of the International Conference on Computer Design*, 1997.

[29] D. Patterson and J. Hennessey. *Computer Architecture: A Quantitative Approach*. Morgan Kaufman, 2nd edition, 1996.

[30] D. Quass, A. Gupta, I. Mumick, and J. Widom. Making views self-maintainable for data warehousing. In *Proc. of PDIS'96*, 1996.

[31] E. Riedel, G. Gibson, and C. Faloutsos. Active storage for large scale data mining and multimedia applications. In *Proceedings of 24th Conference on Very Large Databases*, 1998.

[32] Seagate Technology Inc. *The Cheetah 9LP Family: ST39102 Product Manual*, July 1998. Publication number 83329240 Rev B.

[33] The 3Com SuperStack II Switch Datasheets. *http://www.3com.com/products/dsheets/400260a.html*, 1998.

[34] T. Tamura, M. Oguchi, and M. Kitsuregawa. Parallel database processing on a 100 node pc cluster: Cases for decision support query processing and data mining. In *Supercomputing Conference (SC '97)*, 1997.

[35] The Tolly Group. 3Com Corporation Superstack II 9300 Gigabit Ethernet Performance. Available off *http://www.tolly.com*, Jan 1998.

[36] M. Uysal, A. Acharya, R. Bennett, and J. Saltz. A customizable simulator for workstation clusters. In *Proc. of the 11th International Parallel Processing Symposium*, pages 249–54, 1997.

[37] M. Uysal, A. Acharya, and J. Saltz. Structure and performance of decision support algorithms on active disks. Technical Report TRCS98-28, Dept of Computer Science, University of California, Santa Barbara, Oct 1998.

[38] R. Wang, T. Anderson, and D. Patterson. Virtual log-based file systems for a programmable disk. In *Proc. of OSDI'99*, 1999.

[39] John Wilkes. DataMesh research project, phase 1. In *USENIX File System Workshop Proceedings*, pages 63–70, 1992.

[40] R. Winter and K. Auerbach. Giants walk the earth: the 1997 VLDB survey. *Database Programming and Design*, 10(9), Sep 1997.

[41] R. Winter and K. Auerbach. The big time: the 1998 VLDB survey. *Database Programming and Design*, 11(8), Aug 1998.

[42] M. Zaki, S. Parthasarathy, and W. Li. A localized algorithm for parallel association mining. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1997.