# Data Mining on an OLTP System (Nearly) for Free

Erik Riedel[1], Christos Faloutsos,
Greg Ganger, David Nagle

June 1999
CMU-CS-99-151

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

{riedel,christos,ganger,nagle}@cs.cmu.edu

## Abstract

*This paper proposes a scheme for scheduling disk requests that takes advantage of the ability of high-level functions to operate directly at individual disk drives. We show that such a scheme makes it possible to support a Data Mining workload on an OLTP system almost for free: there is only a small impact on the throughput and response time of the existing workload. Specifically, we show that an OLTP system has the disk resources to provide a consistent one third of its sequential bandwidth to a background Data Mining task with close to zero impact on OLTP throughput and response time at high transaction loads. At low transaction loads, we show much lower impact than observed in previous work. This means that a production OLTP system can be used for Data Mining tasks without the expense of a second dedicated system. Our scheme takes advantage of close interaction with the on-disk scheduler by reading blocks for the Data Mining workload as the disk head "passes over" them while satisfying demand blocks from the OLTP request stream. We show that this scheme provides a consistent level of throughput for the background workload even at very high foreground loads. Such a scheme is of most benefit in combination with an Active Disk environment that allows the background Data Mining application to also take advantage of the processing power and memory available directly on the disk drives.*

*1. Department of Electrical and Computer Engineering*

This research is sponsored by DARPA/ITO through ARPA Order D306, and issued by Indian Head Division, NSWC under contract N00174-96-0002. We are indebted to generous contributions from the member companies of the Parallel Data Consortium. At the time of this writing, these companies include Hewlett-Packard Laboratories, Symbios Logic, Data General, Compaq, Intel, 3Com, Quantum, IBM, Seagate Technology, Hitachi, Siemens, Novell, Wind River Systems, and Storage Technology Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any supporting organization or the U.S. Government.

**ACM Computing Reviews Keywords:** B.4.2 Input/output devices, H.2.8 Database applications, C.3.0 Special-purpose and application-based systems, B.4 Input/Output and Data Communications.

## 1. Introduction

Query processing in a database system requires several resources, including 1) memory, 2) processor cycles, 3) interconnect bandwidth, and 4) disk bandwidth. Performing additional tasks, such as data mining, on a transaction processing system without impacting the existing workload would require there to be "idle" resources in each of these four categories. A system that uses Active Disks [Riedel98] provides additional memory and compute resources at the disk drives that are not utilized by the transaction processing workload. Using Active Disks to perform highly-selective scan and aggregation operations by computing directly at the drives keeps the interconnect requirements low. This leaves the disk arm and media as the critical resources. This paper proposes a scheduling algorithm at the disks that allows a background sequential workload to be satisfied essentially for free while servicing random foreground requests. We start with a simple priority-based scheduling scheme that allows the background workload to proceed with a small impact on the foreground work and then extend this system to read additional blocks completely "for free". We also show that these benefits are consistent at high foreground transaction loads and as data is striped over a larger number of disks.

## 2. Background and Motivation

The use of data mining to elicit patterns from large databases is becoming increasingly popular over a wide range of application domains and datasets [Fayyad98, Chaudhuri97, Widom95]. One of the major obstacles to starting a data mining project within an organization is the high initial cost of purchasing the necessary hardware. This means that someone must "take a chance" on the up front investment simply on the suspicion that there may be interesting "nuggets" to be mined from the organizations existing databases.

The most common strategy for data mining on a set of transaction data is to purchase a second database system, copy the transaction records from the OLTP system to the decision support system each evening, and perform mining tasks only on the second system, i.e. to use a "data warehouse" separate from the production system. This strategy not only requires the expense of a second system, but requires the management cost of maintaining two complete copies of the data. Table 1 compares a transaction system and a decision support system from the same manufac-

| system | # of CPUs | memory (GB) | # of disks | storage (GB) | live data (GB) | cost ($) |
|---|---|---|---|---|---|---|
| NCR WorldMark 4400 (TPC-C) | 4 | 4 | 203 | 1,822 | 1,400 | $839,284 |
| NCR TeraData 5120 (TPC-D 300) | 104 | 26 | 624 | 2,690 | 300 | $12,269,156 |

Table 1: Comparison of an OLTP and a DSS system from the same vendor. Data from *www.tpc.org*, May and June 1998.

turer. The decision support system contains a larger amount of compute power, and higher aggregate I/O bandwidth, even for a significantly smaller amount of live data. In this paper, we argue that the ability to operate close to the disk makes it possible for a significant amount of data mining to be performed using the transaction processing system, without requiring a second system at all. This provides an effective way for an organization to "bootstrap" its mining activities.

Active Disks provide an architecture to take advantage of the processing power and memory resources available in future generation disk drives to perform application-level functions. Next generation drive control chips have processing rates of 150 and 200 MHz and use standard RISC cores, with the promise of up to 500 MIPS processors in two years [Cirrus98, TriCore98]. This makes it possible to perform computation directly on commodity disk drives, offloading server systems and network resources by computing at the edges of the system. The core advantages of this architecture are 1) the parallelism in large systems, 2) the reduction in interconnect bandwidth requirements by filtering and aggregating data directly at the storage devices, before it is placed onto the interconnect and, and 3) closer integration with on-disk scheduling and optimization. Figure 1 illustrates the architecture of such a system. Previous work has shown that selective and

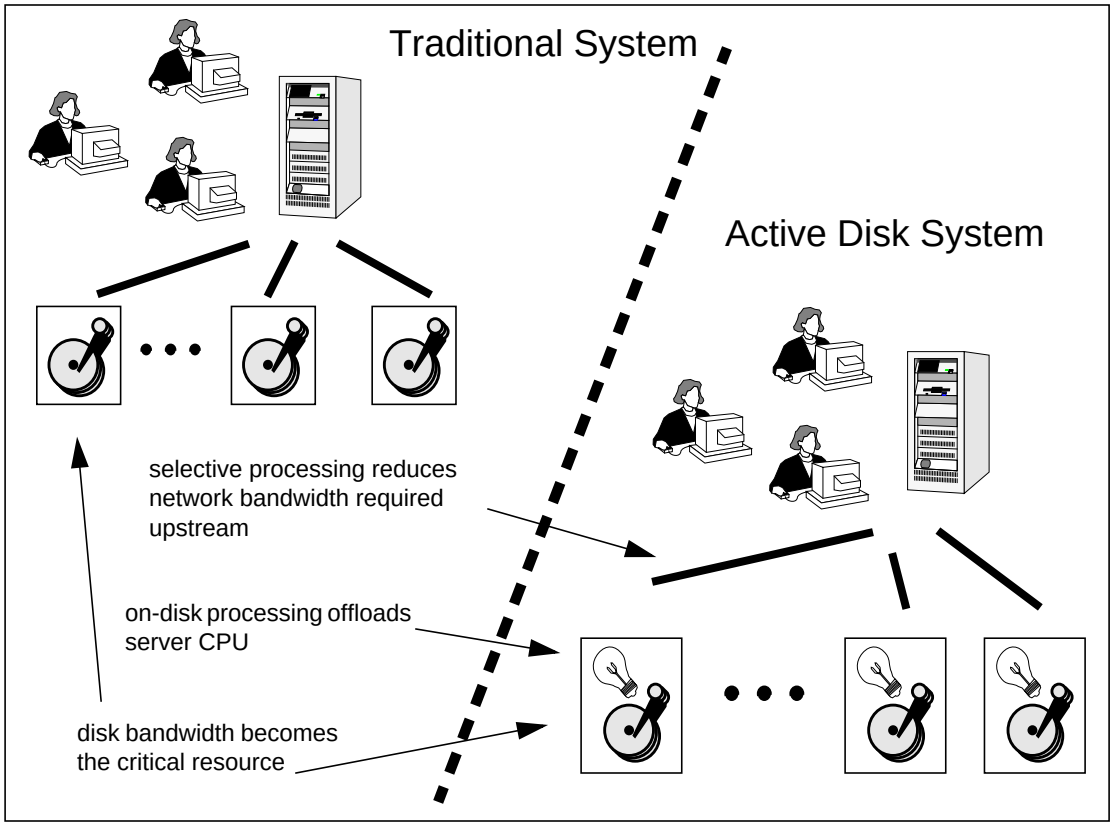



Figure 1: Diagram of a traditional server and an Active Disk architecture. By moving processing to the disks, the amount of data transferred on the network is reduced, the computation can take advantage of the parallelism provided by the disks and benefit from closer integration with on-disk scheduling. This allows the system to continue to support the same transaction workload with additional mining functions operating at the disks.

highly parallel operations such as aggregation, selection, or selective joins can be offloaded to Active Disks or similar systems [Riedel98, Acharya98, Keeton98]. Many data mining operations including nearest neighbor search, association rules [Agrawal96], ratio and singular value decomposition [Korn98], and clustering [Zhang97, Guha98] eventually translate into a few large sequential scans of the entire data. If these selective, parallel scans can be performed directly at the individual disks, then the limiting factor will be the bandwidth available for reading data from the disk media.

## 3. Proposed System

The performance benefits of Active Disks are most dramatic with the highly-selective parallel scans that form a core part of many data mining applications. The scheduling system we propose assumes that a mining application can be specified abstractly as:

```
(1)  foreach block(B) in relation(X)
(2)       filter(B) -> B'
(3)  combine(B') -> result(Y)
```

*assumption*: ordering of blocks does not affect the result of the computation

where steps (1) and (2) can be performed directly at the disk drives in parallel, and step (3) combines the results from all the disks at the host once the individual computations complete.

The performance of an application that fits this model and has a low computation cost for the `filter` function and high selectivity (data reduction from B to B') will be limited by the raw bandwidth available for sequential reads from the disk media. In a dedicated mining system, this bandwidth would be the full sequential bandwidth of the individual disks. However, even in a system running a transaction processing workload, a significant amount of the necessary bandwidth is available in the "idle" time between and during disk seek and rotational latency for the transaction workload.

The key insight is that during disk seeks for a foreground transaction processing (OLTP) workload, disk blocks passing under the disk head can be read "for free". If the blocks are useful to a background application, they can be read without any impact on the OLTP response time by completely hiding the read within the request's rotational delay. In other words, while the disk is moving to the requested block, it opportunistically reads blocks that it passes over and provides them to the data mining application. If this application is operating directly at the disk drive in an Active Disk environment, then the block can be immediately processed, without ever having to be transferred to the host. As long as the data mining application - or any other background application - can issue a large number of requests at once and does not depend on the order of processing the requested background blocks, the background application will read a significant portion of its data without any cost to the OLTP workload. The disk will ensure that only blocks of a particular application-specific size (e.g. database pages) are provided, and that all the blocks requested are read exactly once, but the order of blocks will be determined by pattern of the OLTP requests.

Figure 2 shows the basic intuition of the proposed scheme. The drive maintains two request queues: 1) a queue of demand foreground requests that are satisfied as soon as possible; and 2) a list of the background blocks that are satisfied when convenient. Whenever the disk plans a seek to satisfy a request from the foreground queue, it checks if any of the blocks in the background queue are "in the path" from the current location of the disk head to the desired foreground request. This is accomplished by comparing the delay that will be incurred by a direct seek and rotational latency at the destination to the time required to seek to an alternate location, read some number of blocks and then perform a second seek to the desired cylinder. If this "detour" is

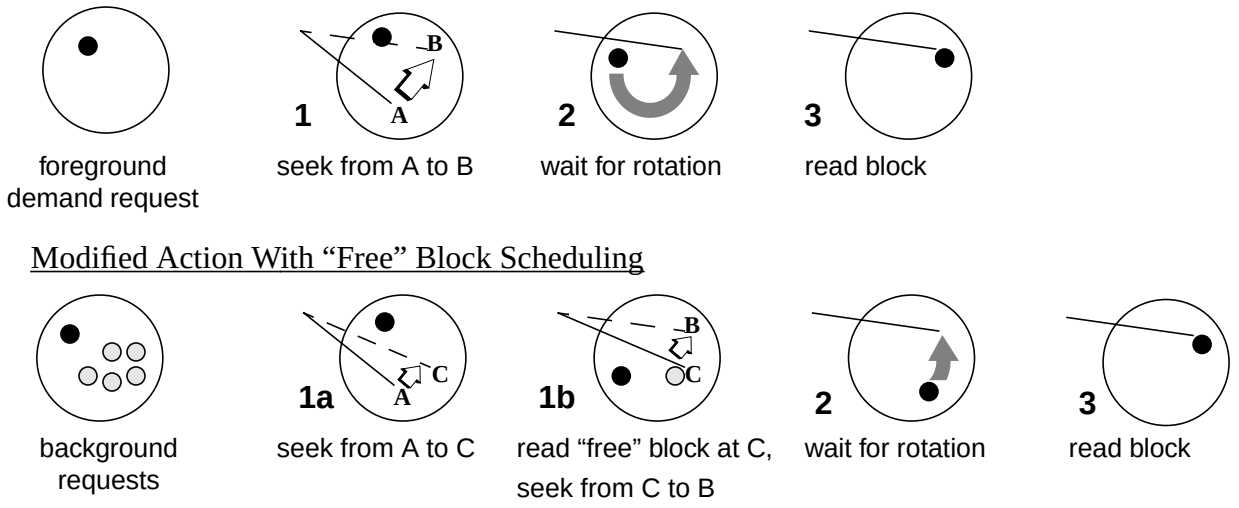



Figure 2: Illustration of 'free' block scheduling. In the original operation, a request to read or write a block causes the disk to seek from its current location (A) to the destination cylinder (B). It then waits for the requested block to rotate underneath the head. In the modified system, the disk has a set of potential blocks that it can read "at its convenience". When planning a seek from A to B, the disk will consider how long the rotational delay at the destination will be and, if there is sufficient time, will plan a shorter seek to C, read a block from the list of background requests, and then continue the seek to B. This additional read is completely 'free' because the time waiting for the rotation to complete at cylinder B is completely wasted in the original operation.

shorter than the rotational delay, then some number of background blocks can be read without increasing the response time of the foreground request. If multiple blocks satisfy this criterion, the location that satisfies the largest number of background blocks is chosen. Note that in the simplest case, the drive will continue to read blocks at the current location, or seek to the destination and read some number of blocks before the desired block rotates under the head.

## 4. Experiments

All of our experiments were conducted using a detailed disk simulator [Ganger98], synthetic traces based on simple workload characteristics, and traces taken from a server running a TPC-C transaction workload. The simulation models a closed system with a think time of 30 milliseconds which approximates that seen in our traces. We vary the multiprogramming level of the OLTP workload to illustrate increasing foreground load on the system. Multiprogramming level is specified in terms of disk requests, so a multiprogramming level of 10 means that there are ten disk requests active in the system at any given point (either queued at one of the disks or waiting in think time).

In the synthetic workloads, the OLTP requests are evenly spaced across the entire surface of the disk with a read to write ratio of 2:1 and a request size that is a multiple of 4 kilobytes chosen from an exponential distribution with a mean of 8 kilobytes. The background data mining (Mining) requests are large sequential reads with a minimum block size of 8 kilobytes. In the experiments, Mining is assumed to occur across the entire database, so the background workload reads

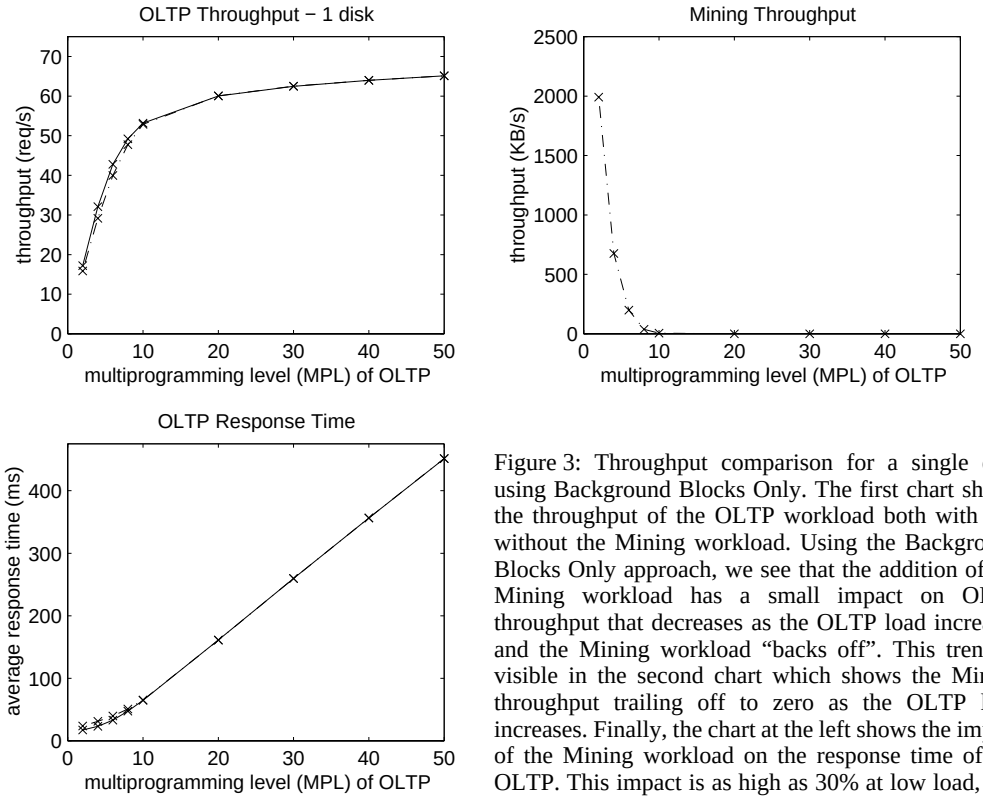



Figure 3: Throughput comparison for a single disk using Background Blocks Only. The first chart shows the throughput of the OLTP workload both with and without the Mining workload. Using the Background Blocks Only approach, we see that the addition of the Mining workload has a small impact on OLTP throughput that decreases as the OLTP load increases and the Mining workload "backs off". This trend is visible in the second chart which shows the Mining throughput trailing off to zero as the OLTP load increases. Finally, the chart at the left shows the impact of the Mining workload on the response time of the OLTP. This impact is as high as 30% at low load, and decreases to zero as the load increases.

the entire surface of the disk. Reading the entire disk is a pessimistic assumption and further optimizations are possible if only a portion of the disk contains data (see Section 4.5).

All simulations run for one hour of simulated time and complete between 50,000 and 250,000 foreground disk requests and up to 900,000 background requests, depending on the load.

There are several different approaches for integrating a background sequential workload with the foreground OLTP requests. The simplest only performs background requests during disk idle times (i.e. when the queue of foreground requests is completely empty). The second uses the "free blocks" technique described above to read extra background blocks during the rotational delay of an OLTP request, but does nothing during disk idle times. Finally, a scheme that integrates both of these approaches allows the drive to service background requests whenever they do not interfere with the OLTP workload. This section presents results for each of these three approaches followed by results that show the effect is consistent as data is striped over larger numbers of disks. Finally, we present results for the traced workload that correspond well with those seen for the synthetic workload.

### *4.1. Background Blocks Only, Single Disk*

Figure 3 shows the performance of the OLTP and Mining workloads running concurrently as the OLTP load increases. Mining requests are handled at low priority and are serviced only when

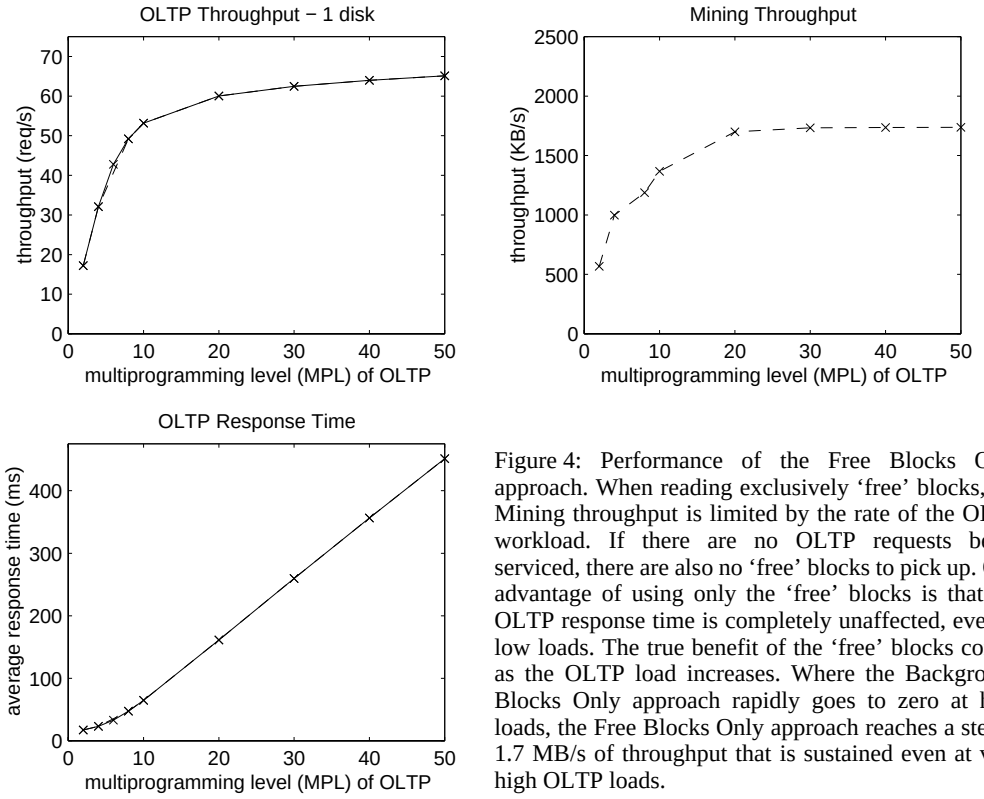



Figure 4: Performance of the Free Blocks Only approach. When reading exclusively 'free' blocks, the Mining throughput is limited by the rate of the OLTP workload. If there are no OLTP requests being serviced, there are also no 'free' blocks to pick up. One advantage of using only the 'free' blocks is that the OLTP response time is completely unaffected, even at low loads. The true benefit of the 'free' blocks comes as the OLTP load increases. Where the Background Blocks Only approach rapidly goes to zero at high loads, the Free Blocks Only approach reaches a steady 1.7 MB/s of throughput that is sustained even at very high OLTP loads.

the foreground queue is empty. The first chart shows that increasing the OLTP load increases throughput until the disk saturates and queues begin to build. This effect is also clear in the response time chart below, where times grow quickly at higher loads. The second chart shows the throughput of the Mining workload at about 2 MB/s for low load, but decreases rapidly as the OLTP load increases, forcing out the low priority background requests. The third chart shows the impact of Mining requests on OLTP response time. At low load, when requests are already fast, the OLTP response time increases by 25 to 30%. This increase occurs because new OLTP requests arrive while a Mining request is being serviced. As the load increases, OLTP request queueing grows, reducing the chance that an OLTP request would wait behind a Mining request in service and eliminating the increase in OLTP response time as the Mining work is forced out.

### *4.2. 'Free' Blocks Only, Single Disk*

Figure 4 shows the effect of reading 'free' blocks while the drive performs seeks for OLTP requests. Low OLTP loads produce low Mining throughput because little opportunity exists to exploit 'free' block on OLTP requests. As the foreground load increases, the opportunity to read 'free' blocks improves, increasing Mining throughput to about 1.7 MB/s. This is a similar level of throughput seen in the Background Blocks Only approach, but occurs under high OLTP load where the first approach could sustain significant Mining throughput only under light load, rapidly dropping to zero for loads above 10. Since Mining does not make requests during completely

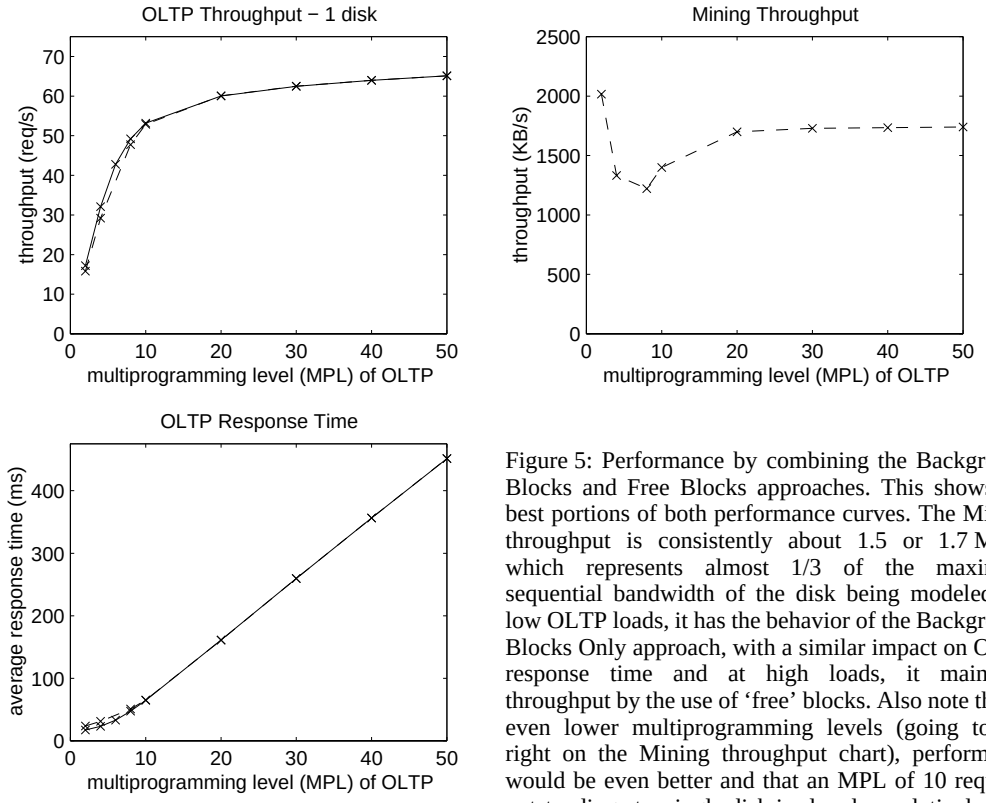



Figure 5: Performance by combining the Background Blocks and Free Blocks approaches. This shows the best portions of both performance curves. The Mining throughput is consistently about 1.5 or 1.7 MB/s, which represents almost 1/3 of the maximum sequential bandwidth of the disk being modeled. At low OLTP loads, it has the behavior of the Background Blocks Only approach, with a similar impact on OLTP response time and at high loads, it maintains throughput by the use of 'free' blocks. Also note that at even lower multiprogramming levels (going to the right on the Mining throughput chart), performance would be even better and that an MPL of 10 requests outstanding at a single disk is already a relatively high absolute load.

idle time in the 'Free' Blocks Only approach, OLTP response time does not increase at all. The only shortcoming of the 'Free' Blocks Only approach is the low Mining throughput under light OLTP load.

### *4.3. Combination of Background and 'Free' Blocks, Single Disk*

Figure 5 shows the effect of combining these two approaches. On each seek caused by an OLTP request, the disk reads a number of 'free' blocks as described in Figure 2 in the previous section. This models the behavior of a query that wishes to scan a large portion of the disk, but does not care in which order the blocks are processed. Full table scans in the TPC-D queries, aggregations, or the association rule discovery application [Riedel98] could all make use of this functionality. Figure 5 shows that Mining throughput increases to between 1.4 and 2.0 MB/s at low load. At high loads, when the Background Blocks Only approach drops to zero, the combined system continues to provide a consist throughput at about 2.0 MB/s without any impact on OLTP throughput or response time. The full sequential bandwidth of the modeled disk (if there were no foreground requests) is only 5.3 MB/s to read the entire disk[1], so this represents more than 1/3 of the raw bandwidth of the drive completely "in the background" of the OLTP load.

1. As mentioned before, reading the entire disk is pessimistic since reading the inner tracks of modern disk drives is significantly slower than reading the outer tracks. If we only read the beginning of the disk (which is how "maximum bandwidth" numbers are determined in spec sheets), the bandwidth would be as high as 6.6 MB/s, but our scheme would also perform proportionally better.

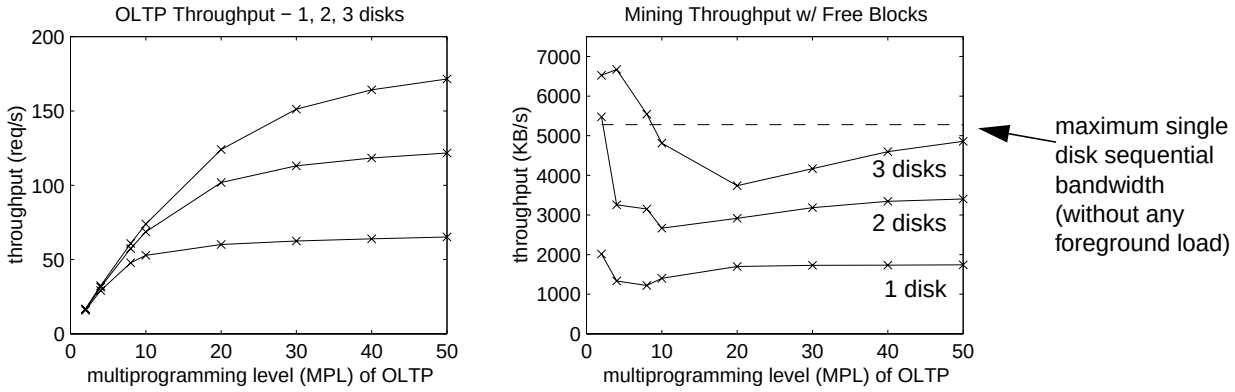



Figure 6: Throughput of 'free' blocks as additional disks are used for the same OLTP workload. If we stripe the same amount of data over a larger number of disks while maintaining a constant OLTP load, we see that the total Mining throughput increases as expected.

### *4.4. Combination Background and 'Free' Blocks, Multiple Disks*

Systems optimized for bandwidth rather than operations per second will usually have more disks than strictly required to store the database (as illustrated by the decision support system of Table 1). This same design choice can be made in a combined OLTP/Mining system.

Figure 6 shows that Mining throughput using our scheme increases linearly as the workloads are striped across a multiple disks. Using two disks to store the same database (i.e. increasing the number of disks used to store the data in order to get higher Mining throughput, while maintaining the same OLTP load and total amount of "live" data) provides a Mining throughput above 50% of the maximum drive bandwidth across all load factors, and Mining throughput reaches more than 80% of maximum with three disks.

We can see that the performance of the multiple disk systems is a straightforward "shift" of the single disk results, where the Mining throughput with *n* disks at a particular MPL is simply *n* times the performance of a single disk at *1/n* that MPL. The two disk system at 20 MPL performs twice as fast as the single disk at 10 MPL, and similarly with 3 disks at 30 MPL. This predictable scaling in Mining throughput as disks are added bodes well for database administrators and capacity planners designing these hybrid systems. Additional experiments indicate that these benefits are also resilient in the face of load imbalances ("hot spots") in the foreground workload.

### *4.5. 'Free' Blocks, Details*

Figure 7 shows the performance of the 'free' block system at a single, medium foreground load (an MPL of 10 as shown in the previous charts). The rate of handling background requests drops steadily as the fraction of unread background blocks decreases and more and more of the unread blocks are at the "edges" of the disk (i.e. the areas not often accessed by the OLTP workload and the areas that are expensive to seek to). This means that if data can be kept near the "front" or "middle" of the disk, overall 'free' block performance would improve (staying to the right of the second chart in Figure 7). Extending our scheduling scheme to "realize" when only a

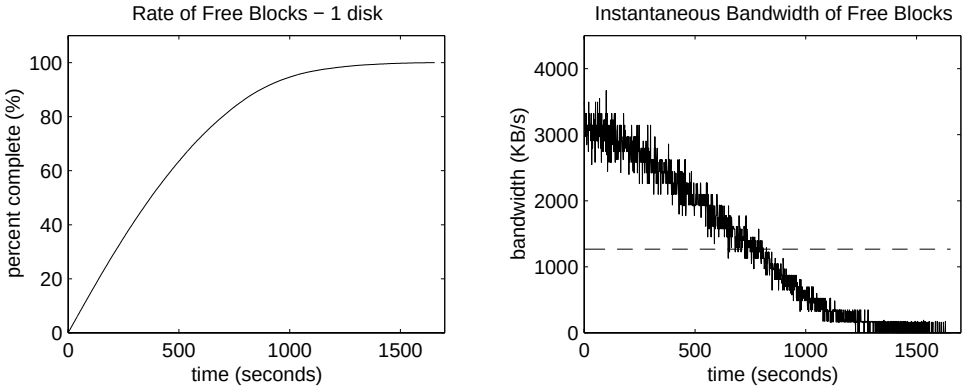



Figure 7: Details of ‘free’ block throughput with a particular foreground load. The first plot shows the amount of time needed to read the entire disk in the background at a multiprogramming level of 10. The second plot shows the instantaneous bandwidth of the background workload over time. We see that the bandwidth is significantly higher at the beginning, when there are more background blocks to choose from. As the number of blocks still needed falls, less of them are “within reach” of the ‘free’ algorithm and the throughput decreases. The dashed line shows the average bandwidth of the entire operation.

small portion of the background work remains and issue some of these background requests at normal priority (with the corresponding impact on foreground response time) should also improve overall throughput. The challenge is to find an appropriate trade-off of impact on the foreground against improved background performance.

Finally, note that even with the basic scheme as described here, it is possible to read the entire 2 GB disk for ‘free’ in about 1700 seconds (under 28 minutes), allowing a disk to perform over 50 “scans per day” [Gray97] of its entire contents completely unnoticed.

### *4.6. Workload Validation*

Figure 8 shows the results of a series of traces taken from a real system running TPC-C with varying loads. The traced system is a 300 MHz Pentium II with 128 MB of memory running Win-

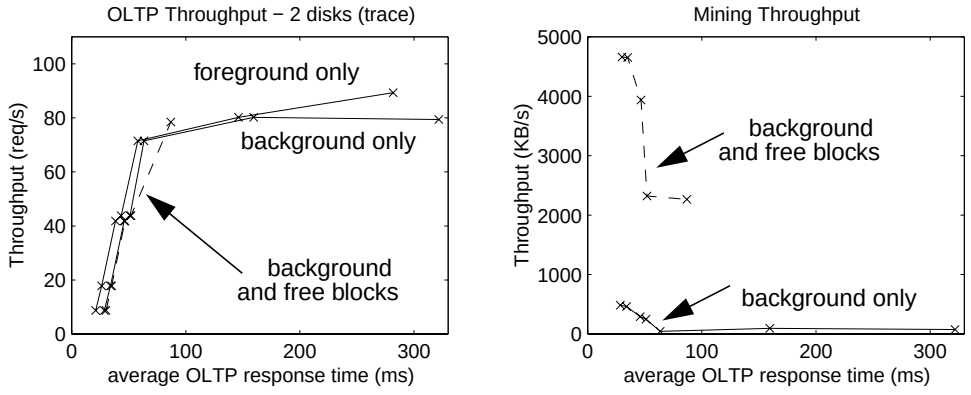



Figure 8: Performance for the traced OLTP workload in a two disk system. The numbers are more variable than the synthetic workload, but the basic benefit of the ‘free’ block approach is clear. We see that use of the ‘free’ block system provides a significant boost above use of the Background Blocks Only approach. Note that since we do not control the multiprogramming level of the traced workload, the x axes in these charts are the average OLTP response time, which combines the three charts given in the earlier figures into two and makes the MPL a hidden parameter.

dows NT and Microsoft SQL Server on a one gigabyte TPC-C test database striped across two Viking disks. When we add a background sequential workload to this system, we see results similar to those of the synthetic workloads. At low loads, several MB/s of Mining throughput are possible, with a 25% impact on the OLTP response time. At higher OLTP loads, the Mining workload is forced out, and the impact on response time is reduced unless the 'free' block approach is used. The Mining throughput is a bit lower than the synthetic workload shown in Figure 6, but this is most likely because the OLTP workload is not evenly spread across the disk while the Mining workload still tries to read the entire disk.

The disk being simulated and the disk used in the traced system is a 2.2 GB Quantum Viking 7,200 RPM disk with a (rated) average seek time of 8 ms. We have validated the simulator against the drive itself and found that read requests come within 5% for most of the requests and that writes are consistently under-predicted by an average of 20%. Extraction of disk parameters is a notoriously complex job [Worthington95], so a 5% difference is a quite reasonable result. The under-prediction for writes could be the result of several factors and we are looking in more detail at the disk parameters to determine the cause of the mismatch. It is possible that this is due to a more aggressive write buffering scheme modeled in the simulator than actually exists at the drive. This discrepancy should have only a minor impact on the results presented here, since the focus is on seeks and reads, and an underprediction of service time would be pessimistic to our results. The demerit figure [Ruemmler94] for the simulation is 37% for all requests.

## 5. Discussion

Previous work [Riedel98] has shown that Active Disks - individual disk drives that provide application-level programmability - can provide the compute power, memory, and reduction in interconnect bandwidth to make data mining queries efficient on a system designed for a less demanding workload. This paper illustrates that there is also sufficient disk bandwidth in such a system to make a combined transaction processing and data mining workload possible. We show that a significant amount of data mining work can be accomplished with only a small impact on the existing transaction processing performance. This means that if the "dumb" disks in a traditional system are replaced with Active Disks, there will be sufficient resources in compute power, memory, interconnect bandwidth, and disk bandwidth to support both workloads. It is no longer necessary to buy an expensive second system with which to perform decision support and basic data mining queries.

At the very least, one could design a backup system would be able to read the entire contents of a 2 GB disk in 30 minutes without any impact on the running OLTP workload. It is no longer necessary to run backups in the middle of the night, stop the system in order to back it up, or endure reduced performance during backups.

The results in Section 4.5 indicate that our current scheme is pessimistic because it requires the background workload to read every last block on the disk, even at much lower bandwidth. There are a number of optimization in data placement and the choice of which background blocks to "go after" to be explored, but our simple scheme shows that significant gains are possible.

## 6. Related Work

Previous studies of combined OLTP and decision support workloads on the same system indicate that the disk is the critical resource [Paulin97]. Paulin observes that both CPU and memory utilization is much higher for the Mining workload than the OLTP, which is also clear from the design of the decision support system shown in Table 1 in our introduction. In his experiments, all system resources are shared among the OLTP and decision support workloads with an impact of 36%, 70%, and 118% on OLTP response time when running decision support queries against a heavy, medium, and light transaction workload, respectively. The author concludes that the primary performance issue in a mixed workload is the handling of I/O demands on the data disks, and suggests that a priority scheme is required in the database system as a whole to balance the two types of workloads.

Brown, Carey and DeWitt [Brown92, Brown93] discuss the allocation of memory as the critical resource in a mixed workload environment. They introduce a system with multiple workload classes, each with varying response time goals that are specified to the memory allocator. They show that a modified memory manager is able to successfully meet these goals in the steady state using ‘hints’ in a modified LRU scheme. The modified allocator works by monitoring the response time of each class and adjusting the relative amount of memory allocated to a class that is operating below or above its goals. The scheduling scheme we propose here for disk resources also takes advantage of multiple workload classes with different structures and performance goals. In order to properly support a mixed workload, a database system must manage all system resources and coordinate performance among them.

Existing work on disk scheduling algorithms [Denning67, ..., Worthington94] shows that dramatic performance gains are possible by dynamically reordering requests in a disk queue. One of the results in this work indicates that many scheduling algorithms can be performed equally well at the host [Worthington94]. The scheme that we propose here takes advantage of additional flexibility in the workload (the fact that requests for the background workload can be handled at low priority and out of order) to expand the scope of reordering possible in the disk queue. Our scheme also requires detailed knowledge of the performance characteristics of the disk (including exact seek times and overhead costs such as settle time) as well as detailed logical-to-physical mapping information to determine which blocks can be picked up for free. This means that this scheme would be difficult, if not impossible, to implement at the host without close feedback on the current state of the disk mechanism. This makes it a compelling use of additional “smarts” directly at the disk.

With the advent of Storage Area Networks (SANs), storage devices are being shared among multiple hosts performing different workloads [HP98, Seagate98, Veritas99]. As the amount and variety of sharing increases, the only central location to optimize scheduling across multiple workloads will be directly on the devices themselves.

## 7. Conclusions

This paper presents a scheduling scheme that takes advantage of the properties of large, scan-intensive workloads such as data mining to extract additional performance from a system that already seems completely busy. We used a detailed disk simulator and both synthetic and traced workloads to show that there is sufficient disk bandwidth to support a background data mining workload on a system designed for transaction processing. We propose to take advantage of 'free' blocks that can be read during the seeks required by the OLTP workload. Our results indicate that we can get one third of the maximum sequential bandwidth of a disk for the background workload without any effect on the OLTP response times. This level of performance is possible even at high transaction loads. At low transaction loads, it is possible to achieve an even higher level of background throughput if we allow a small impact (between 25 and 30% impact on transaction response time) on the OLTP performance.

The use of such a scheme in combination with Active Disks that also provide parallel computational power directly at the disks makes it possible to perform a significant amount of data mining without having to purchase a second, dedicated decision support system or maintain two copies of the data.

## 8. Bibliography


[Acharya98] Acharya, A., Uysal, M. and Saltz, J. "Active Disks" *ASPLOS*, October 1998.

[Agrawal96] Agrawal, R. and Schafer, J. "Parallel Mining of Association Rules" *IEEE Transactions on Knowledge and Data Engineering* 8,6. December 1996.

[Brown92] Brown, K., Carey, M., DeWitt, D., Mehta, M. and Naughton, J. "Resource Allocation and Scheduling for Mixed Database Workloads" Technical Report, University of Wisconsin, 1992.

[Brown93] Brown, K., Carey, M. and Livny, M. "Managing Memory to Meet Multiclass Workload Response Time Goals" VLDB, August 1993.

[Chaudhuri97] Chaudhuri, S. and Dayal, U. "An Overview of Data Warehousing and OLAP Technology" *ACM SIGMOD Record*, March 1997.

[Cirrus98] Cirrus Logic, Inc. "New Open-Processor Platform Enables Cost-Effective, System-on-a-chip Solutions for Hard Disk Drives" *www.cirrus.com/3ci*, June 1998.

[Denning67] Denning, P.J. "Effects of Scheduling on File Memory Operations" *AFIPS Spring Joint Computer Conference*, April 1967.

[Fayyad98] Fayyad, U. "Taming the Giants and the Monsters: Mining Large Databases for Nuggets of Knowledge" *Database Programming and Design*, March 1998.

[Ganger98] Ganger, G.R., Worthington, B.L. and Patt, Y.N. "The DiskSim Simulation Environment Version 1.0 Reference Manual" *Technical Report*, University of Michigan, February 1998.

[Gray97] Gray, J. "What Happens When Processing, Storage, and Bandwidth are Free and Infinite?" *IOPADS Keynote*, November 1997.

[Guha98] Guha, S., Rastogi, R. and Shim, K. "CURE: An Efficient Clustering Algorithm for Large Databases" *SIGMOD*, 1998.

[HP98] Hewlett-Packard Company "HP to Deliver Enterprise-Class Storage Area Network Management Solution" *www.openview.hp.com/press/press/press.asp?docid=74*, October 1998.

[Keeton98] Keeton, K., Patterson, D.A. and Hellerstein, J.M. "A Case for Intelligent Disks (IDISKs)" *SIGMOD Record* 27 (3), August 1998.

[Korn98] Korn, F., Labrinidis, A., Kotidis, Y. and Faloutsos, C. "Ratio Rules: A New Paradigm for Fast, Quantifiable Data Mining" *VLDB*, August 1998.

[Paulin97] Paulin, J. "Performance Evaluation of Concurrent OLTP and DSS Workloads in a Single Database System" Master's Thesis, Carleton University, November 1997.

[Riedel98] Riedel, E., Gibson, G. and Faloutsos, C. "Active Storage For Large-Scale Data Mining and Multimedia" *VLDB*, August 1998.

[Ruemmler94] Ruemmler, C. and Wilkes, J. "An Introduction to Disk Drive Modeling" *IEEE Computer* 27 (3), March 1994.

[Seagate98] Seagate Technology "Storage Networking" *www.seagate.com/corp/vpr/literature/papers/sn.shtml*, November 1998.

[TriCore98] TriCore News Release "Siemens Announced Availability of TriCore-1 For New Embedded System Designs" *www.tri-core.com*, March 1998.

[Veritas99] Veritas Corporation "Storage Area Networks" *www.veritas.com/newtech/san/index.html*, June 1999.

[Widom95] Widom, J. "Research Problems in Data Warehousing" *CIKM*, November 1995.

[Worthington94] Worthington, B.L., Ganger, G.R. and Patt, Y.N. "Scheduling Algorithms for Modern Disk Drives" *SIGMETRICS*, May 1994.

[Worthington95] Worthington, B.L., Ganger, G.R., Patt, Y.N., Wilkes, J. "On-Line Extraction of SCSI Disk Drive Parameters" *SIGMETRICS*, May 1995.

[Zhang97] Zhang, T., Ramakrishnan, R. and Livny, M. "BIRCH: A New Data Clustering Algorithm and Its Applications" *Data Mining and Knowledge Discovery* 1 (2), 1997.