Array Access Bounds For Block Storage Memory Systems

Arvin Park

K. Balasubramanian

Richard J. Lipton

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

CS-TR-042-86

July 1, 1986

Abstract

This paper explores array storage and access strategies on block storage devices (interleaved memories, RAM disks, and disk drives). A tradeoff is exhibited for row access speed and column access speed, and an optimal upper bound for their product is established. Practical array access strategies are discussed as well as extensions and further research.

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and by the Office of Naval Research under Contracts Nos. N00014-85-C-0446 and N00014-85-K-0465, and by the National Science Foundation under Cooperative Agreement No. DCR-8420948. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

1. Introduction

Paging performance can be a dominating factor in an application's running time [1]. Many seemingly efficient data structures and algorithms lose orders of magnitude in performance because they generate an excessive number of page faults. Several papers have been written on array storage schemes in paged memory systems [2][3][4]. McKellar and Coffman [2] noted the performance advantages of storing submatricies on pages instead of the standard row or column major formats (Figure 1). Fischer and Probert [3] have developed efficient methods of reorganizing

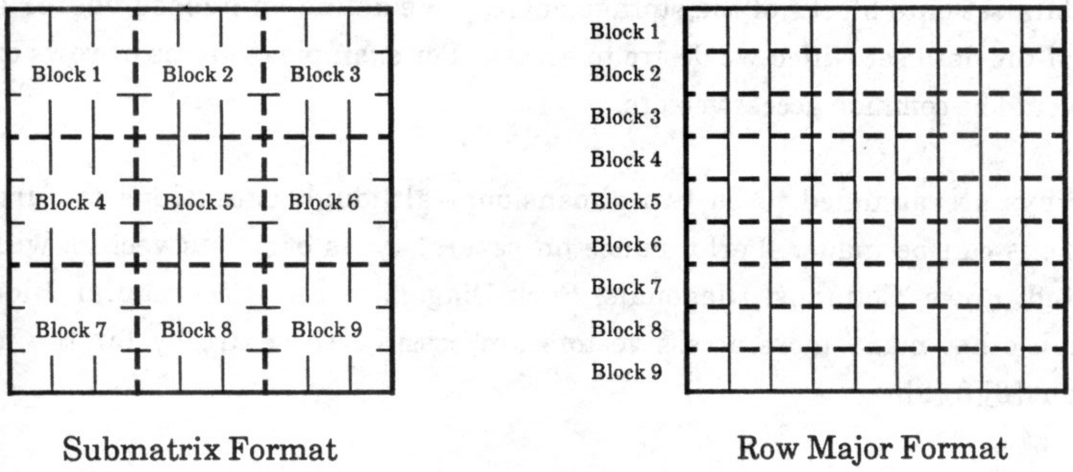

Figure 1

a row or column oriented matrix into submatrix form. This paper extends these results by showing that tradeoffs exist between average row access speed S_r (which is defined as number of row elements retrieved divided by the number of blocks accessed) and average column access speed S_c (defined similarly). We prove the $S_r S_c$ product is optimally upper bounded by the block size N. Practical array access strategies are developed and extensions to these results are discussed.

2. Model

Block storage devices span an entire range of block sizes and access times: Cache lines and interleaved semiconductor stores support block sizes from one to several dozens of words and have access times measured in hundreds of nanoseconds; Rotating storage supports block sizes of hundreds to thousands of words and access times in the tens of milliseconds; RAM disks, CCD, and magnetic bubble stores fit between these two extremes in block sizes and access times.

To analyze array storage schemes for all of these different systems we have developed a simple model. We assume a block memory system is a set of L storage "blocks" of exactly N data elements (words). We will measure performance by the number of "page pulls" required by an application to perform its task.

We define **data set** as the entire collection of data elements (words) which is stored in the memory system. We define an **allocation** as an assignment of elements of the data set onto blocks of the storage device. We define an **access vector** as a subset of the data set which we desire to access. For example columns or rows of an array would be common access vectors.

Arrays are assumed to be two dimensional although extensions to further dimensions can be made. Performance on several types of access vectors will be evaluated; Rows, Columns, Diagonals, Back-Diagonals, and Rectangular Blocks. Although other many more access vectors are possible these are by far the most prevalent [6][8][9].

Allocations

To achieve maximal performance it is necessary to allocate the array elements to blocks of the storage device minimizing the total number of blocks referenced on a set of access vectors. This is difficult even if the set of access vectors are known in advance. For a fixed allocation, the number of vectors of a given size which fit exactly onto the minimum number of blocks is a great deal smaller than the number of possible vectors.

Assume an access vector is of size V, and that N (the number of elements in a block) divides V. The entire data set consists of KN elements and fits neatly into K blocks of the storage device. For a fixed allocation there exists

$$\binom{K}{V/N}$$

possible access vectors of size V that fit exactly into K blocks. This is in contrast to

$$\binom{KN}{V}$$

possible vectors of size V. The ratio of these two quantities is:

$$\frac{\binom{K}{V/N}}{\binom{KN}{V}} = \frac{\dfrac{K!}{(K-V/N)!(V/N)!}}{\dfrac{KN!}{(KN-V)!\,V!}} = \frac{K!\,V!\,(KN-V)!}{KN!\,(V/N)!\,(K-V/N)!}$$

Using Stirling's Approximation this becomes:

$$\approx \frac{(2\pi K)^{1/2}\left(\frac{K}{e}\right)^{K}(2\pi V)^{1/2}\left(\frac{V}{e}\right)^{V}(2\pi(KN-V))^{1/2}\left(\frac{KN-V}{e}\right)^{(KN-V)}}{(2\pi KN)^{1/2}\left(\frac{KN}{e}\right)^{KN}(2\pi V/N)^{1/2}\left(\frac{V/N}{e}\right)^{V/N}(2\pi(K-V/N))^{1/2}\left(\frac{K-V/N}{e}\right)^{(K-V/N)}}$$

$$= N^{(1/2+V/N-V)}\,K^{(-KN+1)}\,V^{(V-V/N)}\,(K-V/N)^{(KN-V-K+V/N)} \approx \left(\frac{V}{KN}\right)^{V}$$

$$= \left(\frac{\mathit{SIZE\ OF\ VECTOR}}{\mathit{SIZE\ OF\ DATA\ SET}}\right)^{(\mathit{SIZE\ OF\ VECTOR})}$$

This is a very small number if the data set is much larger than the desired vector.

Row-Column Bound

There is a tradeoff between facility in accessing rows and in accessing columns. For example, if an array is stored in row major form (blocks contain adjacent row elements) rows can be accessed very quickly. For each block accessed all the elements can be used. At the same time column access becomes slow (For each block accessed only one element can be used). This tradeoff can be bounded. We define **average row access speed** as an average over all rows in the array, of the number of elements in a row divided by the number of blocks which contain elements of the row. If a row was of size N and was contained entirely on one block of size N, the access speed for that row would be N. If the same row was contained on N separate blocks the row access speed would be 1. **Average column access speed** is defined analogously.

For any two dimensional array A that is partitioned into blocks of size N without replication of elements, i.e., each element is stored in one and only one block (it should be obvious that storing the same element twice in the same block cannot be of any advantage), the average row access speed S_r multiplied by the average column access speed S_c is less than or equal to N. $S_r S_c \geqq N$. This not only applies to rows and columns, but any two **orthogonal partitions** of the array: A **partition** K of a data set D, is a set of disjoint subsets of D that spans D (an individual disjoint subset is called a **member** of K). Two partitions A,B of a data set D are said to be **orthogonal** iff $\forall a_i \in A \wedge \forall b_j \in B, a_i \cap b_j = \{d\}, d \in D$ or $a_i \cap b_j = \emptyset$, and $a_i \cap b_j = a_k \cap b_l \Leftrightarrow i{=}k$ and $j = l$. Two partitions, A and B, will be called **completely orthogonal** iff they are orthogonal and $\forall a_i \in A \wedge \forall b_j \in B, \mid a_i \cap b_j \mid = 1$. We now establish Theorem 1.

Theorem 1: Given any two orthogonal partitions A, B of a data set D and an allocation of D onto blocks of size N without replication. The average partition access speed on A 'S_A' multiplied by the average partition access speed on B 'S_B' is bounded by N. $S_A S_B \leqq N$.

We first establish two lemmas:

Lemma 1: $\forall\, x,y \in Z^+,\ x/y + y/x \geqq 2$

Proof:

$$(x-y)^2 \geq 0 \ \Rightarrow\ x^2 - 2xy + y^2 \geq 0 \ \Rightarrow\ x^2 + y^2 \geq 2xy \ \Rightarrow\ \frac{x^2+y^2}{xy} \geq 2 \ \Rightarrow\ \frac{x}{y} + \frac{y}{x} \geq 2$$

Lemma 2: For any set of p positive integers x_i, ($i = 1,2,3 \ldots p$)

$$\sum_{i=1}^{p}\sum_{j=1}^{p}\frac{x_j}{x_i} \geq p^2$$

Proof:

$$\sum_{i=1}^{p}\sum_{j=1}^{p}\frac{x_j}{x_i} = \frac{1}{2}\left(2\sum_{i=1}^{p}\sum_{j=1}^{p}\frac{x_j}{x_i}\right) = \frac{1}{2}\left(\sum_{i=1}^{p}\sum_{j=1}^{p}\frac{x_j}{x_i} + \frac{x_i}{x_j}\right)$$

By Lemma one:

$$\geq \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p} 2 = \frac{1}{2}\,(2)\,p^2 = p^2$$

We can now proceed with the proof of the theorem. Let M be the number of blocks in data set D, where each block is of size N. Let A and B (e.g., rows and columns) be two orthogonal partitions of D. Each block i of data set D intersects r_i different members of A (rows) and c_i different members of B (columns). An intersection between a block i and a partition element of A is called an **A-fragment of i**. Let $k_i = N/r_i$ and $j_i = N/c_i$.

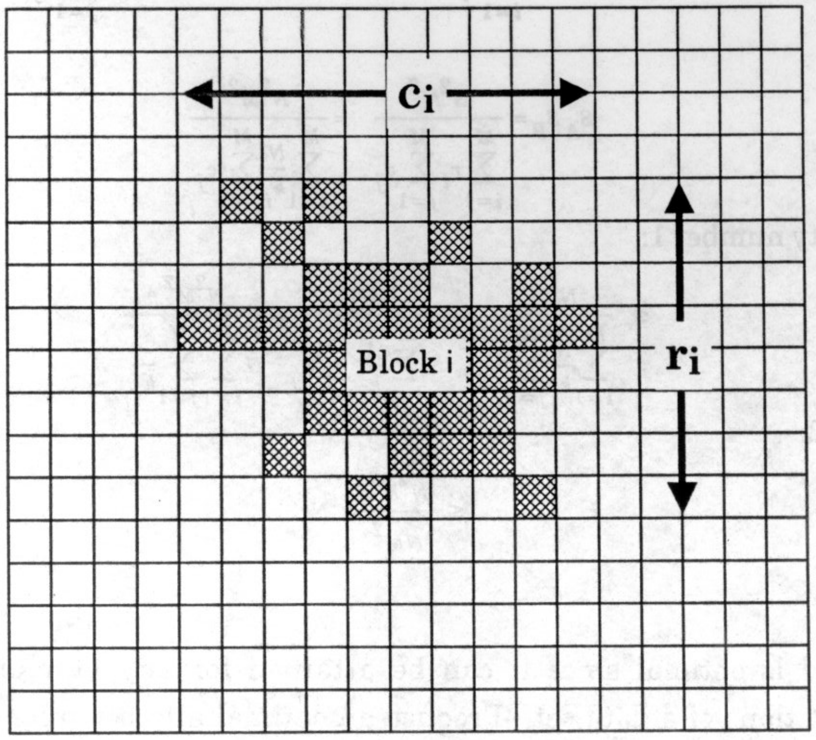

Figure 2

The orthogonality property of A and B, implies the following:

$$r_i c_i \geq N$$

Suppose this were not true:

$$r_i c_i < N$$

The sum of all of the r_i A-fragments of a block i equals N, this implies that the average fragment is of size greater than c_i. By orthogonality each A-fragment and B-fragment can intersect in at most one element. So block i must intersect more than c_i members of B. Contradiction! We now establish two inequalities.

$$r_i c_i \geq N \Rightarrow N/r_i \leq c_i \Rightarrow k_i \leq c_i \qquad (1)$$

$$r_i c_i \geq N \Rightarrow N/c_i \leq r_i \Rightarrow j_i \leq r_i \qquad (2)$$

We now proceed with the main proof.

$$S_A = \frac{\textit{Data Set Size}}{\textit{Number of Fragments}} = \frac{NM}{\sum_{i=1}^{M} r_i} \qquad S_B = \frac{\textit{Data Set Size}}{\textit{Number of Fragments}} = \frac{NM}{\sum_{j=1}^{M} c_j}$$

$$S_A S_B = \frac{N^2M^2}{\sum_{i=1}^{M} r_i \sum_{j=1}^{M} c_j} = \frac{N^2M^2}{\sum_{i=1}^{M} \frac{N}{k_i} \sum_{j=1}^{M} c_{j_i}}$$

Using inequality number 1:

$$\leq \frac{N^2M^2}{\sum_{i=1}^{M} \frac{N}{k_i} \sum_{j=1}^{M} k_j} = \frac{N^2M^2}{N \sum_{i=1}^{M} \frac{1}{k_i} \sum_{j=1}^{M} k_j} = \frac{N^2M^2}{N \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{k_j}{k_{i_i}}}$$

Using Lemma 2:

$$\leq \frac{N^2M^2}{NM^2} = N$$

Done.

This bound is optimal since it can be attained for any pair of completely orthogonal partitions of a data set. Produce an optimal allocation for partitions A and B of data set D in the following manner. Find a number $C < N$ that divides N. Divide the members (rows) of A into disjoint groups of size N/C. Call the set of these groups A^*. Divide the members (columns) of B into disjoint groups of size C. Call the set of these groups B^*. For each pair (x,y), where $x \in A^*$ and $y \in B^*$, Produce a block $G_{x,y}$ which has the following N elements.

$$G_{x,y} = \bigcup^{\forall i \in x \wedge \forall j \in y} i \cap j$$

Now each member of A that we want to access is distributed in fragments of size N/C across different blocks of the allocation. Members of B are similarly distributed in fragments of size C. The A access speed "S_A" is therefore $S_A = N/C$, and the B access speed, $S_B = C$. The product of these two, $S_A S_B = N$, which meets the upper bound of Theorem 1. This upper bound is therefore optimal.

Rectangular blocks achieve this upper bound for the row-column tradeoff. Various diamonds and parallelograms achieve the bound for combinations of rows, diagonals, columns, and back-diagonals (Figure 3). Diagonals and back diagonals intersect differently than the other combinations of rows, columns, diagonals, and back-

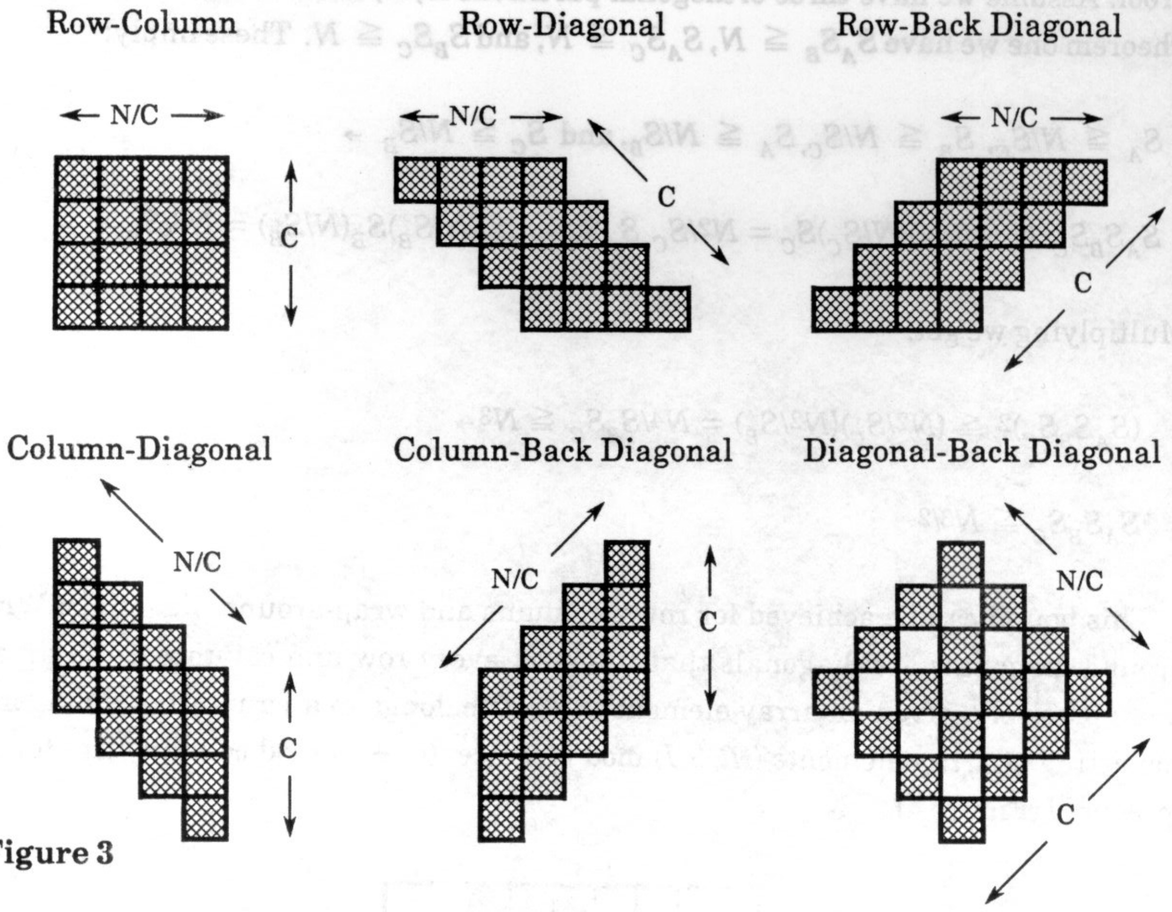

Figure 3

diagonals. This is because a given diagonal does not intersect every back diagonal it crosses. A matrix is divided into two disjoint sets of diagonals and back diagonals that intersect like the red and black squares of a checker board.

Three Way Access Bounds

How does this bound extend to accesses along three different partitions of the array? (for instance rows, columns, and diagonals) A bound can be arrived at by noting that each pair of access vectors must be constrained by the pairwise bound $S_A S_B \leqq N$. This gives us Theorem 2,

<u>**Theorem 2:**</u> Given any any three orthogonal partitions A, B, and C, of a data set D, and an allocation of D onto blocks of size N without replication. The product of the average partition access speeds on A, B, and C in bounded by $N^{3/2}$. $S_A S_B S_C \leqq N^{3/2}$.

Proof: Assume we have three orthogonal partitions A, B, and C of data set D. From Theorem one we have $S_AS_B \leqq N$, $S_AS_C \leqq N$, and $S_BS_C \leqq N$. These imply:

$$S_A \leqq N/S_C,\ S_B \leqq N/S_C, S_A \leqq N/S_B, \text{ and } S_C \leqq N/S_B \rightarrow$$

$$S_AS_BS_C \leqq (N/S_C)(N/S_C)S_C = N^2/S_C, S_AS_BS_C \leqq (N/S_B)S_B(N/S_B) = N^2/S_B \rightarrow$$

Multiplying we get:

$$(S_AS_BS_C)^2 \leqq (N^2/S_C)(N^2/S_B) = N^4/S_BS_C \leqq N^3 \rightarrow$$

$$S_AS_BS_C \leqq N^{3/2}$$

This bound can be achieved for rows, columns and wrap-around diagonals. Wrap around diagonals are diagonals that intersect every row and column of the array. More precisely, given an array element (R,C), it belongs to a wrap-around diagonal consisting of array elements $((R + I) \bmod \text{row size}, (C + I) \bmod \text{column size})$ for all integers I (Figure 4).

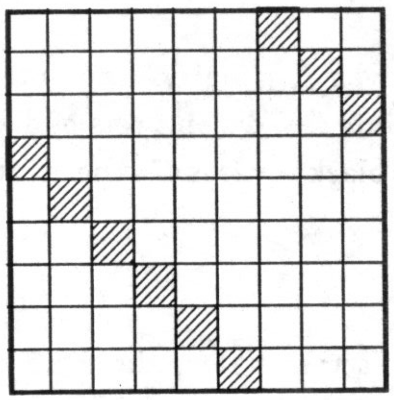

Figure 4 A Wrap Around Diagonal

The bound can be achieved for an M X M array with block sizes N (assume $N^{1/2}$ divides M). Divide the array into N submatricies of size $(M/N^{1/2})$ X $(M/N^{1/2})$. Now construct blocks of size N by taking one element from the same position in each of the N submatricies of the array. This will yield M^2/N blocks of size N. Each row, column, or back diagonal of size M is contained on $M/N^{1/2}$ different blocks. Hence each row, column, and back-diagonal has access speed $N^{1/2}$. The product of the three access speeds is then $N^{3/2}$ which meets the upper bound of Theorem 2. Figure 5 illustrates the construction for the case where $M = 9$ and $N = 9$.

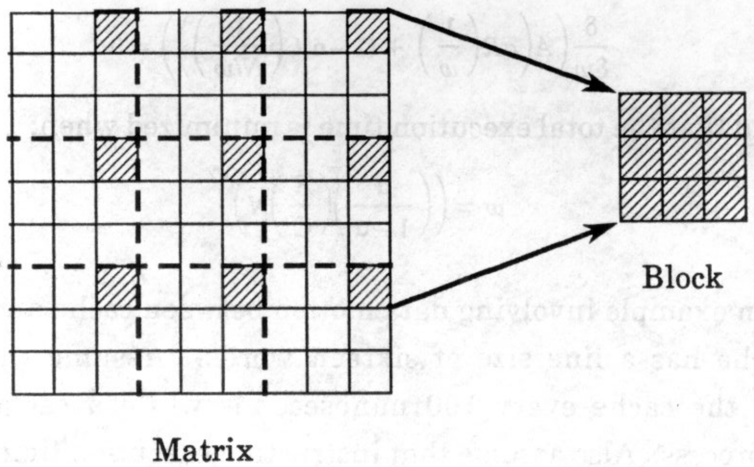

Figure 5

We have not been able to achieve the $N^{3/2}$ bound for conventional diagonals, rows and columns. $N^{1/2}$ X $N^{1/2}$ Submatrix blocks achieve a $N^{3/2}/2$ product for diagonal, row, and column access speeds which is within a factor of two of the upper bound in Theorem 2. We conjecture that the $N^{3/2}$ product cannot be achieved for row, column, and diagonal access speeds. This bound will be the focus of future research.

Array Access Strategies

The row-column speed bound suggests strategies for array storage if the relative frequencies of row and column accesses are known beforehand. For example, if a particular application accesses random rows and columns. The row accesses occur with probability α and column accesses occur with probability $1 - \alpha$. We can produce an allocation that minimizes the total execution time.

Let the row size be $R >> N$ and column size be $C >> N$. N is the block size. We will allocate rectangular sections of the array onto different blocks. Assume these blocks are of width w and length N/w ($1 \leqq w \leqq N$). Let A be the total number of accesses. The total execution time is then given by the expression.

$$A\left(\alpha R\left(\frac{1}{w}\right) + (1-\alpha)C\left(\frac{1}{N/w}\right)\right)$$

We minimize this by taking the derivative with respect to w and setting the resulting expression equal to zero.

$$\frac{\delta}{\delta w}\left(A\left(\alpha R\left(\frac{1}{w}\right) + (1-\alpha)C\left(\frac{1}{N/w}\right)\right)\right) = 0$$

From this we find that the total execution time is minimized when:

$$w = \left(\left(\frac{\alpha}{1-\alpha}\right)\left(\frac{R}{C}\right)N\right)^{1/2}$$

Let us take an example involving data motion between cache and main memory. suppose the cache has a line size of sixteen words. Assume that data can be referenced from the cache every 100 nanoseconds while a cache fault takes 1 microsecond to process. Also assume that instruction execution time is insignificant (this is not a bad assumption if most of the code resides in a separate instruction cache). Suppose rows and columns are of equal length, and that rows are referenced half of the time and columns are referenced half of the time. Using the formula for w above, we see that the blocks should be of width four and length four. Let's compare this storage scheme to the conventional row major scheme.

Average access time per word (row major)
= 1/2(avg. word access time for row) + 1/2(avg. word access time for column)
= 1/2((1/16)1000ns + (15/16)100ns) + 1/2(1000ns) = **578.125ns**

Average access time per word (4X4 blocks)
= 1/2(avg. word access time for row) + 1/2(avg. word access time for column)
= 1/2((1/4)1000ns + (3/4)100ns) + 1/2((1/4)1000ns + (3/4)100ns) = **325ns**

The new storage scheme gained a factor of two in performance! Of course the row column access patterns become more complicated, and this leads to more complicated coding or compilation, but in some applications this is the only way to achieve optimal performance.

The benefits of this type of storage become even more pronounced as the block sizes increase. In computations that require numerous accesses to RAM disk, CCD memory, or rotating storage devices even larger gains can be realized.

A note on diagonals

It should be noted that diagonals/back-diagonals are not completely orthogonal to rows/columns even though (see figure 3) it seems that we have a parallelogram shape that seems to tile the plane and achieve the bound of N for the product of row-diagonal speedups. This is because the arrays we are dealing with are rectangular so diagonals and back-diagonals do not fit "neatly" along the boundaries of the array. These 'edge effects' lead to a speedup somewhat smaller than that which can be achieved for rows and columns. These edge effects diminish in importance as the arrays get larger , and in fact for arrays that extend to infinity on all four sides, rows and diagonals are *completely* orthogonal ! However even with rows (& columns) being completely orthogonal to diagonals (back-diagonals) for 'infinite' arrays, it still does not seem possible to achieve more than $N^{3/2}/2$ for the product of row-column-diagonal speedup. It is also noteworthy that diagonals and back-diagonals do not form a pair of completely orthogonal partitions of the array elements but rather *two pairs of orthogonal partitions of two disjoint subsets of the array elements!* This seems to indicate some basic structural differences between rows/columns and diagonals/back-diagonals which needs to be investigated further.

Conclusions and Future Research

We have shown that the product of access speeds for any two of (row, column, diagonal, back-diagonal) is optimally upper bounded by the block size N. This was extended to show that the product any three access speeds (row, column, diagonal, back-diagonal) is upper bounded by $N^{3/2}$. This three way bound can be met for rows, columns, and wrap around diagonals, but it remains and open question if this bound can be met for rows columns and regular diagonals.

Future research will involve extending these bounds to multidimensional arrays and examining the advantages of redundant storage (storing a single array element on more than one block).

References

[1] A. Park. H. Garcia-Molina, "Performance Through Massive Memory", Internal Report, Department of Computer Science, Princeton University, June 1986.

[2] A. C. McKellar, E. G. Coffman, Jr., "Organizing Matricies and Matrix Operations for Paged Memory Systems", Communications of the ACM, Volume 12, Number 3, March 1969.

[3] P. C. Fischer, R. L. Probert, "Storage Reorganization Techniques for Matrix Computation in a Paging Environment", Communications of the ACM, Volume 22, Number 7, July 1979.

[4] C. B. Moler, "Matrix Computations with Fortran and Paging", Communications of the ACM, Volume 15, Number 4, April 1972.

[5] K. E. Batcher, "The Multidimensional Access Memory in STARAN," IEEE Transactions on Computers, Volume C-26, Number 2, pp. 174-177, February 1977.

[6] P. Budnick, D. J. Kuck, "The Organization and Use of Parallel Memories", IEEE Transactions on Computers, Volume C-20, Number 12, pp. 1566-1569, December 1971.

[7] T. Feng, "Data Manipulating Functions in Parallel Processors and Their Implementations", IEEE Transactions on Computers, Volume C-23, Number 3, pp. 309-318, March 1974.

[8] D. H. Lawrie, "Access and Alignment of Data in an Array Processor," IEEE Transactions on Computers, Volume C-24, Number 12, pp. 1145-1155, December 1975.

[9] D. H. Lawrie, "The Prime Memory System for Array Access," IEEE Transactions on Computers, Volume C-31, Number 5, pp. 134-141, May 1982.

[10] C. V. Ravi, "On the Bandwidth and Interference in Interleaved Memory Systems," IEEE Transactions on Computers, Volume C-21, Number 8, August 1972.