

**CENTER FOR
PARALLEL OPTIMIZATION**

OPTIMAL TILINGS FOR PARALLEL DATABASE DESIGN

by

Jonathan Yackel and Robert R. Meyer

Computer Sciences Technical Report #1046

September 1991

Optimal Tilings for Parallel Database Design*

Jonathan Yackel[†] Robert R. Meyer[†]

Abstract

The computing time benefits of parallelism in database systems (achieved by using multiple processors to execute a query) must be weighed against communication, startup, and termination overhead costs that increase as a function of the number of processors used. We consider problems of allocating data among the processors so as to optimize the execution time for certain type of queries. We present lower bounds for these combinatorial problems and demonstrate how processors may be optimally assigned by “tiling” the partitioned data grid with optimal configurations.

1 Introduction

In highly-parallel database machines (e.g., Gamma [2], Bubba [1], Non-Stop SQL [11], XPRS [10] and Volcano [6]) relations are “partitioned” (see Livny *et al* [8] and Ries and Epstein [9] for early partitioning strategies) across multiple processors. This allows each processor to execute a portion of a query in parallel with the other processors, resulting in a lower response time for the query. However, there is communication overhead associated with initiating and terminating a query on multiple processors, and this overhead increases as a function of the number of processors used to execute

*This research was partially supported by the Air Force Office of Scientific Research under grant 89-0410, and by the National Science Foundation under grants CCR-8907671 and DCR-8512862.

[†]Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706.

a query. (This overhead is primarily in the form of additional messages to control the execution of the query on additional processors and, in the Gamma database machine [2], increases linearly with the number of employed processors.) In order to balance the workload among the processors, Multi-Attribute GrId deClustering (MAGIC), as introduced by Ghandeharizadeh [3], partitions a relation by assigning ranges of several attribute values to each processor in the system. To illustrate MAGIC consider the partitioning of the Employee relation **EMP** in figure 1. For parallel computation,

		Salary in \$K		
		0-20	20-50	> 50
Age in Years	0-25	1	1	2
	26-50	1	3	3
	> 50	2	3	2

Figure 1: Processor assignment for the EMP relation

MAGIC partitions the **EMP** relation by establishing ranges of **Salary** and **Age** attribute values. Each cell of this grid corresponds to a fragment of the relation and must be assigned to some processor. For example, the cell which contains records with **Salary** attribute values that range from 0 to 20 and **Age** attribute values that range from 26 to 50 is assigned to processor 1. Given a query on either the **Age** or **Salary** attribute, the predicate of the query maps to either a row or a column (termed a “slice”) of the grid and the corresponding processors are used to execute it. Note that, for the assignment depicted by figure 1, every processor is assigned three cells, and every query requires two processors.

Although our principal focus is on the limiting case in which the goal is to minimize query overhead (measured via non-convex fixed-charge functions that count the number of distinct processors in slices), the optimal processor assignments that we thereby obtain also minimize some more complex response time functions. For example, suppose the response time r for a query as a function of the number of processors ν used by the query is modeled by $r(\nu) = \mathcal{O}^1\nu + Q^1/\nu$, where \mathcal{O}^1 is the overhead associated with a single processor and Q^1 is the processing time for a query on a single processor, *i.e.* the query overhead increases proportionally with the number of processors used for the query, while the processing time is inversely proportional to the number of processors¹. As we will show in section 5.3, under certain assumptions the solutions that minimize overhead are also optimal for the response time function $r(\nu)$.

1.1 Overview

This paper formalizes problems of optimally assigning the cells of a multidimensional grid to a given number of processors. In section 2 we present a mathematical statement of versions of the problem. In section 3, we develop lower bounds on the optimal value. Section 4 develops optimal configurations of cells for individual processors, and section 5 provides combinations of these configurations producing optimal assignments

¹The Gamma database machine results presented in [2] justify this assertion.

that attain the lower bounds. Our conclusions and future research directions are contained in section 6.

2 Problem Statement

Suppose that we wish to assign the cells of a D -dimensional grid to N processors, and that the size of the grid is $M_1 \times M_2 \times \dots \times M_D$ (i.e., the d th attribute is partitioned into M_d ranges). Let $V := \prod_d M_d$ denote the number of cells (volume) of the grid. A “slice” is a $(D - 1)$ -dimensional subgrid containing all the cells with a common value for a given coordinate (this corresponds to a query). For example, in an $M_1 \times M_2$ grid the slices are the M_1 rows and the M_2 columns, and in an $M_1 \times M_2 \times M_3$ grid the slices are the $M_1 + M_2 + M_3$ two-dimensional subgrids. Let \mathcal{S} denote the collection of slices.

Given an assignment of cells to processors and an arbitrary slice s of the grid, let ν_s denote the number of *distinct* processors in the slice s (this may be thought of as the “chromatic index” of slice s). Given a processor p , let the “load” L_p denote the number of cells assigned to p . (From a geometric viewpoint, L_p may be thought of as the area or volume occupied by processor p .) The objective functions for the optimization problems that we develop measure total or worst case overhead. Let $\theta_{\text{total}} := \sum_{s \in \mathcal{S}} \nu_s$ and $\theta_{\text{max}} := \max_{s \in \mathcal{S}} \nu_s$. Note that if each slice has the same frequency of access and we are interested in minimizing the *average* query overhead, then we should minimize θ_{total} . If, on the other hand, we are interested in minimizing the *worst case* overhead incurred by *any query*, then θ_{max} should be minimized. Load balancing constraints are defined by specifying a load for each processor. In the case of homogeneous processors, we would expect $L_p = \lfloor V/N \rfloor$ or $\lceil V/N \rceil$ for each p , but for non-homogeneous processors loads related to processor power could be specified. The problem, which we now state formally, is to minimize the overhead subject to a fixed load balance:

Let the following data be given: the dimensions of the grid: $M_1 \times M_2 \times \dots \times M_D$, a number of processors N , and a load L_p for each processor.

Find an assignment that

$$\begin{aligned} &\text{minimizes } \theta \quad (\text{where } \theta \text{ is chosen as } \theta_{\text{total}} \text{ or } \theta_{\text{max}}) \\ &\text{s.t. } \quad \text{every cell is assigned to a processor,} \\ &\quad \text{and processor } p \text{ is assigned } L_p \text{ cells } (p = 1, 2, \dots, N). \end{aligned}$$

It is easily seen that the number of assignments satisfying the balancing constraint is

$$\binom{V}{L_1} \binom{V - L_1}{L_2} \binom{V - L_1 - L_2}{L_3} \dots \binom{V - L_1 - L_2 - \dots - L_{D-1}}{L_D} = \frac{V!}{\prod_{p=1}^N (L_p)!}.$$

Complete enumeration of these assignments is not feasible even for relatively small problems. For example, given a 5×5 grid, 5 processors, and a load of 5 for each processor, there are more than 623×10^{12} assignments that satisfy the balancing constraint.

A similar class of data aggregation problems was studied by Helman [7]. He showed the following problem to be NP-complete: Given a set of objects $D = \{d_1, d_2, \dots, d_{kN}\}$ (where N and k are positive integers), a set of “requests” $Q =$

$\{q_1, q_2, \dots, q_m\}$, where each $q_i \subseteq D$, and an integer C , is there a partition P of D into N aggregates, each of size k such that $\sum_{i=1}^m \text{Numb}(P, q_i) \leq C$ where $\text{Numb}(P, q_i)$ is the number of aggregates in P which intersect q_i ? The corresponding optimization problem

$$\begin{aligned} \min \quad & \sum_{i=1}^m \text{Numb}(P, q_i) \\ \text{such that} \quad & P \text{ is a partition of } Q \text{ into } N \text{ aggregates of size } k \end{aligned} \quad (1)$$

is NP-hard.

When the objective function is θ_{total} and a perfect load balance ($L_i = L_j \forall i, j$) is specified, our problem is a restricted form of (1). The set of grid cells in our problem corresponds to Helman's set D , and the set of slices of the grid correspond to the set Q . An assignment of cells to processors with a balanced load corresponds to a partition of D into N equal size aggregates. The numbers of distinct processors in the slices correspond to the $\text{Numb}(P, q_i)$. Since the slices of grids intersect in a regular way, the set Q corresponding to a grid has special structure. Therefore our problem is a restriction of the NP-hard problem (1) where Q has a "grid structure".

3 A Lower Bound on θ_{total}

In this section we show that θ_{total} is equivalently represented as the sum of the " D -perimeters" of the configurations of cells assigned to the individual processors. We develop a lower bound on D -perimeter as a function of load, hence arriving at a lower bound on θ_{total} .

Because reordering the slices of the grid does not affect the objective functions, we may consider the cells assigned to a particular processor in their most "compact" configuration. For a particular configuration of cells, we can permute the slices so that in each dimension the slices containing cells in the configuration are contiguous. For example, Figure 2 shows a configuration of 9 cells with a non-compact and a compact slice ordering.

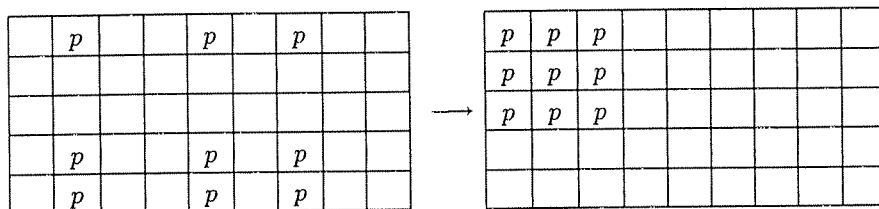
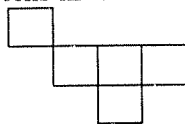


Figure 2: Permuting slices to produce a compact configuration.

Throughout the rest of this paper, the following notation is used. $\mathcal{P}(C)$ denotes the " D -perimeter" of a configuration C of cells. By D -perimeter we mean the sum of the D dimensions of the smallest hyper-rectangle enclosing the cells in their most

compact form. For example, the 2-perimeter of the configuration  is $4 + 3 = 7$. The notation $\mathcal{P}(C_p)$ is used to denote the D -perimeter of the cells assigned

to processor p . Another way to interpret $\mathcal{P}(C_p)$ is as the total number of slices containing processor p .

A key relationship in the development of the lower bounds is

Lemma 1 $\theta_{total} = \sum_{s \in \mathcal{S}} \nu_s = \sum_{p=1}^N \mathcal{P}(C_p)$.

Proof: Given an assignment of grid cells to processors, define

$$\chi_s^p = \begin{cases} 1 & \text{if processor } p \text{ appears in slice } s \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathcal{P}(C_p) = \sum_{s \in \mathcal{S}} \chi_s^p$ and $\nu_s = \sum_p \chi_s^p$, so

$$\theta_{total} = \sum_{s \in \mathcal{S}} \nu_s = \sum_{s \in \mathcal{S}} \sum_p \chi_s^p = \sum_p \sum_{s \in \mathcal{S}} \chi_s^p = \sum_p \mathcal{P}(C_p).$$

■

It is useful to consider the overhead measure θ_{total} from these two different points of view. We first derive results related to a lower bound on the D -perimeter of a configuration of cells.

Theorem 2 For a given D -perimeter \mathcal{P} , let $L^*(\mathcal{P})$ be the maximum load for which D -perimeter \mathcal{P} is achievable. Then

$$L^*(\mathcal{P}) = \left\lceil \frac{\mathcal{P}}{D} \right\rceil^r \left\lfloor \frac{\mathcal{P}}{D} \right\rfloor^{D-r}$$

where $r = \mathcal{P} \bmod D$.

Proof: Let $\mathcal{P}^1, \mathcal{P}^2, \dots, \mathcal{P}^D$ denote the dimensions of a configuration. The maximum load for which D -perimeter \mathcal{P} is achievable is the optimal value of the following problem.

$$L^*(\mathcal{P}) = \max \prod_d \mathcal{P}^d \tag{2}$$

$$\text{s.t.} \quad \sum_d \mathcal{P}^d = \mathcal{P}$$

\mathcal{P}^d a positive integer $\forall d$.

A necessary optimality condition for (2) is that no two \mathcal{P}^d 's differ by more than 1. Furthermore, only one (unordered) set of \mathcal{P}^d 's satisfy this condition:

$$\left\{ \underbrace{\left\lfloor \frac{\mathcal{P}}{D} \right\rfloor, \left\lfloor \frac{\mathcal{P}}{D} \right\rfloor, \dots, \left\lfloor \frac{\mathcal{P}}{D} \right\rfloor}_r, \dots, \underbrace{\left\lceil \frac{\mathcal{P}}{D} \right\rceil, \left\lceil \frac{\mathcal{P}}{D} \right\rceil, \dots, \left\lceil \frac{\mathcal{P}}{D} \right\rceil}_{D-r} \right\}.$$

■

By “inverting” the function $L^*(\mathcal{P})$, we obtain a function $\mathcal{P}^*(L)$ which maps load L to the smallest D -perimeter achievable by a configuration of L cells.

Theorem 3 *The minimum D -perimeter of all configurations of L cells is*

$$\mathcal{P}^*(L) := s \lceil L^{1/D} \rceil + (D - s) \lfloor L^{1/D} \rfloor$$

where s is the smallest positive integer such that

$$\lceil L^{1/D} \rceil^s \lfloor L^{1/D} \rfloor^{D-s} \geq L.$$

Proof: We may bound the D -perimeter of any configuration of L cells from below by finding the smallest D -perimeter \mathcal{P} that satisfies $L^*(\mathcal{P}) \geq L$, since this implies $L > L^*(\mathcal{P} - 1)$, which means that a D -perimeter of $\mathcal{P} - 1$ is not achievable for a load of L .

Consider the following sequence of D -dimensional rectangular blocks.

$$\begin{aligned} Q_0 : & \quad 0 \times 0 \times \dots \times 0 \\ Q_1 : & \quad 1 \times 0 \times \dots \times 0 \\ & \quad \vdots \\ Q_D : & \quad 1 \times 1 \times \dots \times 1 \\ Q_{D+1} : & \quad 2 \times 1 \times \dots \times 1 \\ Q_{D+2} : & \quad 2 \times 2 \times \dots \times 1 \\ & \quad \vdots \\ Q_{2D} : & \quad 2 \times 2 \times \dots \times 2 \\ Q_{2D+1} : & \quad 3 \times 2 \times \dots \times 2 \\ & \quad \vdots \end{aligned}$$

We call these blocks “quasi-hypercubes” since the lengths of any two sides of a block differ by at most 1. Note that the volumes of the quasi-hypercubes in the sequence are strictly increasing after the D th, the volume of the i th quasi-hypercube Q_i for $i \geq D$ is $L^*(i)$, and the D -perimeter for the quasi-hypercubes increases by 1 at each step. The volumes of these quasi-hypercubes are the points at which the lower bound on the D -perimeter increases by 1.

For an arbitrary L , there is a unique smallest quasi-hypercube Q_j whose volume is at least L . Since the volume of Q_j is at least L , by selecting L cells from Q_j a D -perimeter of at most $\mathcal{P}(Q_j)$ is achievable for L . Since the volume of Q_{j-1} is smaller than L , a D -perimeter of $\mathcal{P}(Q_{j-1}) = \mathcal{P}(Q_j) - 1$ is not achievable for L . Therefore the smallest D -perimeter achievable for any configuration of L cells is $\mathcal{P}(Q_j)$. Each dimension of Q_j is either $\lfloor L^{1/D} \rfloor$ or $\lceil L^{1/D} \rceil$ so $\mathcal{P}(Q_j)$ is exactly the D -perimeter bound in the statement of the theorem. \blacksquare

The above argument implies a construction technique for “perimeter-optimal” configurations, *i.e.*, configurations with minimum D -perimeter. An optimal configuration for any L can be constructed by arranging L cells into a partial hypercube as follows. Start with a complete hypercube with sides of length $\lfloor L^{1/D} \rfloor$. Add cells to fill in new $(D - 1)$ -dimensional faces (completing a face before starting on a new one) until the total number of cells is L . The resulting partial hypercube will have sides of length $\lfloor L^{1/D} \rfloor$ and $\lceil L^{1/D} \rceil$, and will measure $\lceil L^{1/D} \rceil$ in as few dimensions as possible. In

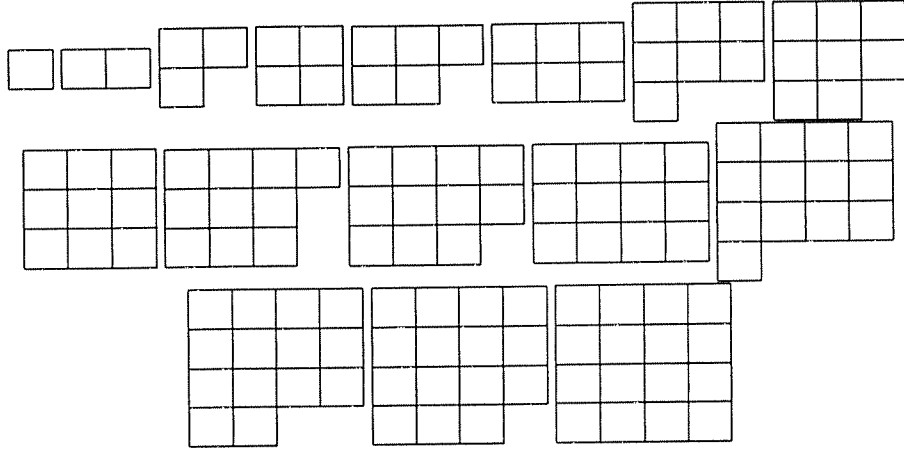


Figure 3: Partial squares with minimum perimeter

min D -perimeter (\mathcal{P}^*)	max load (L^*)			
	2 dimensions	3 dimensions	4 dimensions	5 dimensions
1	—	—	—	—
2	1	—	—	—
3	2	1	—	—
4	4	2	1	—
5	6	4	2	1
6	9	8	4	2
7	12	12	8	4
8	16	18	16	8
9	20	27	24	16
10	25	36	36	32
11	30	48	54	48
12	36	64	81	72
13	42	80	108	108
14	49	100	144	162
15	56	125	192	243

Table 1: Maximum loads and minimum D -perimeters.

figure 3, we show some perimeter-optimal partial squares constructed in this manner with areas ranging from one to sixteen.

Table 1 contains some L^* values for various D -perimeters.

Corollary 4 $\sum_p \mathcal{P}^*(L_p) \leq \theta_{total}.$

Proof: Use theorem 3 and the fact that $\theta_{total} = \sum_p \mathcal{P}(C_p).$ ■

In section 5 we give classes of grids for which the lower bound of corollary 4 is tight.

In the following lemma we present a perimeter optimality test for any configuration of cells.

Lemma 5 *A D -dimensional configuration of L cells with D -perimeter \mathcal{P} has minimum D -perimeter iff*

$$L^*(\mathcal{P} - 1) < L. \quad (3)$$

Proof: If (3) holds, then L is greater than the largest load for which a smaller D -perimeter is achievable. On the other hand if (3) does not hold, then by theorem 3 there is a configuration of L cells with a smaller D -perimeter. ■

In section 3 of Ghandeharizadeh *et al* [5], we derived an alternate bound on \mathcal{P} :

$$\mathcal{P} \geq \lceil D L^{1/D} \rceil.$$

This expression is equivalent to $\mathcal{P}^*(L)$ and therefore is the best possible lower bound when $D = 2$, but for higher dimensions ($D \geq 3$), $\mathcal{P}^*(L)$ provides a better lower bound than $\lceil D L^{1/D} \rceil$ for some values of L .

Theorem 6 *For $D = 2$, $\mathcal{P}^*(L) = \lceil 2L^{1/2} \rceil$.*

Proof: Since \mathcal{P}^* yields the best possible lower bound, we need only show $\mathcal{P}^*(L) \leq \lceil 2L^{1/2} \rceil$.

$$\begin{aligned} \mathcal{P} &\leq \lceil 2L^{1/2} \rceil \\ \iff \mathcal{P} - 2L^{1/2} &< 1 \\ \iff \mathcal{P} - 1 &< 2L^{1/2}. \end{aligned}$$

From lemma 5 we have

$$\begin{aligned} \mathcal{P} &= \mathcal{P}^*(L) \\ \iff L^*(\mathcal{P} - 1) &< L \\ \iff \left\lceil \frac{\mathcal{P} - 1}{2} \right\rceil^r \left\lceil \frac{\mathcal{P} - 1}{2} \right\rceil^{D-r} &< L \end{aligned} \quad (4)$$

where $r = (\mathcal{P} - 1) \bmod 2$.

If $r = 0$ then $\left\lceil \frac{\mathcal{P} - 1}{2} \right\rceil = \frac{\mathcal{P} - 1}{2}$ so (4) becomes

$$\begin{aligned} \left(\frac{\mathcal{P} - 1}{2} \right)^2 &< L \\ \iff \mathcal{P} - 1 &< 2L^{1/2}. \end{aligned}$$

If $r = 1$ then $\lfloor \frac{\mathcal{P}-1}{2} \rfloor = \frac{\mathcal{P}}{2} - 1$ and $\lceil \frac{\mathcal{P}-1}{2} \rceil = \frac{\mathcal{P}}{2}$ so (4) becomes

$$\iff \begin{aligned} \left(\frac{\mathcal{P}}{2} - 1\right) \frac{\mathcal{P}}{2} &< L \\ \mathcal{P}^2 - 2\mathcal{P} &< 4L. \end{aligned}$$

Since both sides of the last inequality are even integers, we may add 1 to the LHS and maintain the inequality.

$$\begin{aligned} \iff \mathcal{P}^2 - 2\mathcal{P} + 1 &< 4L. \\ \iff (\mathcal{P} - 1)^2 &< 4L \\ \iff \mathcal{P} - 1 &< 2L^{1/2}. \end{aligned}$$

■

For $D = 3$, the smallest L for which the two lower bounds are not equal is 37. $\lceil 3(37)^{1/3} \rceil = 10$, but the smallest achievable perimeter is $\mathcal{P}^*(37) = 11$. Note that the two bounds are the same if L is a perfect D -power because $\mathcal{P} = \lceil DL^{1/D} \rceil = DL^{1/D}$ is attainable by a hypercubical configuration in this case.

4 Additional Optimal Configurations

The quasi-hypercubes and related configurations introduced in the proof of theorem 3 are clearly perimeter-optimal. Less square cases will be developed in section 4.1.

4.1 Optimal Two-Dimensional Rectangles

Using the results from the previous section, we can characterize the two-dimensional rectangular blocks which have minimum D -perimeter.

Theorem 7 *An $x \times (x + k)$ or an $(x + k) \times x$ rectangular block is perimeter-optimal iff*

$$\begin{aligned} &k \text{ is even and } 1 + \frac{k}{2}(\frac{k}{2} - 1) \leq x \\ &\text{or} \\ &k \text{ is odd and } 1 + \left(\frac{k-1}{2}\right)^2 \leq x. \end{aligned}$$

Conversely, an $x \times (x + k)$ or an $(x + k) \times x$ rectangle is perimeter-optimal iff the rectangularity increment k is at most

$$\max \left\{ 2 \text{ round}(x^{1/2}), 2 \lfloor x^{1/2} - 1 \rfloor + 1 \right\}$$

where $\text{round}(x)$ rounds x to the nearest integer.

Proof: To prove the first part of the theorem, we simply apply the optimality test. By lemma 5, an $x \times (x + k)$ block is optimal iff

$$\left\lceil \frac{2x + k - 1}{2} \right\rceil^r \left\lfloor \frac{2x + k - 1}{2} \right\rfloor^{D-r} < x^2 + kx \quad (5)$$

where $r = (2x + k - 1) \bmod 2$.

If k is even, (5) reduces to

$$\begin{aligned} & \left\lceil \frac{2x+k-1}{2} \right\rceil \left\lfloor \frac{2x+k-1}{2} \right\rfloor < x^2 + kx \\ \iff & \left(x + \frac{k}{2}\right)\left(x + \frac{k}{2} - 1\right) < x^2 + kx \\ \iff & \frac{k}{2}\left(\frac{k}{2} - 1\right) < x. \end{aligned}$$

The integrality of both sides of the inequality allow us to derive the desired result.

If k is odd, (5) reduces to

$$\begin{aligned} & \left\lfloor \frac{2x+k-1}{2} \right\rfloor^2 < x^2 + kx \\ \iff & \left(x + \frac{k-1}{2}\right)^2 < x^2 + kx \\ \iff & \left(\frac{k-1}{2}\right)^2 < x. \end{aligned}$$

To prove the second part of the theorem, we show that $2 \lfloor (x-1)^{1/2} \rfloor + 1$ and $2 \text{round}(x^{1/2})$ are the largest odd and even integers respectively satisfying (5).

To prove the result for the odd numbers, we start with the expression for x in terms of odd k .

$$\iff \begin{aligned} \left(\frac{k-1}{2}\right)^2 + 1 & \leq x \\ \frac{k-1}{2} & \leq (x-1)^{1/2}. \end{aligned}$$

Since the LHS of the last inequality is integer, we may take the floor of the RHS.

$$\begin{aligned} \iff \frac{k-1}{2} & \leq \lfloor (x-1)^{1/2} \rfloor \\ \iff k & \leq 2 \lfloor (x-1)^{1/2} \rfloor + 1. \end{aligned}$$

Since the RHS of the last inequality is an odd integer, $k = 2 \lfloor (x-1)^{1/2} \rfloor + 1$ is the largest odd integer satisfying (5).

To prove the result for the even numbers we write $x^{1/2}$ in the form $x^{1/2} = r + f$ where r is the integer part and $f \in [0, 1)$ is the fractional part. If $f < \frac{1}{2}$ then $2 \text{round}(x^{1/2}) = 2r$. If $f > \frac{1}{2}$ then $2 \text{round}(x^{1/2}) = 2r + 2$. (For integer x , f is never $\frac{1}{2}$ so $\text{round}(x^{1/2})$ is uniquely defined for integer x .)

If $f < \frac{1}{2}$ and $2 \text{round}(x^{1/2}) = 2r$ then $k = 2r$ satisfies (5) because

$$\frac{k}{2} \left(\frac{k}{2} - 1\right) = r(r-1) = r^2 - r < r^2 + 2fr + f^2 = x$$

and $k = 2r + 2$ violates (5) because

$$\frac{k}{2} \left(\frac{k}{2} - 1\right) + 1 = (r+1)r + 1 = r^2 + r + 1 > r^2 + 2fr + f^2 = x.$$

If $f > \frac{1}{2}$ and $2 \text{round}(x^{1/2}) = 2r + 2$ then $k = 2r + 2$ satisfies (5) because

$$\frac{k}{2} \left(\frac{k}{2} - 1\right) = (r+1)r = r^2 + r < r^2 + 2fr + f^2 = x$$

and $k = 2r + 4$ violates (5) because

$$\frac{k}{2} \left(\frac{k}{2} - 1\right) = (r+2)(r+1) = r^2 + 3r + 2 > r^2 + 2fr + f^2 = x.$$

Therefore $k = 2 \text{ round}(x^{1/2})$ is the largest even integer satisfying (5). ■

Note that the first part of the theorem shows that if a particular two-dimensional rectangle is optimal, then by increasing both dimensions by the same amount, the resulting larger rectangle is also optimal. Figure 4 shows the dimensions of all rectangles with $x \leq 30$ that have minimum perimeter. The integral points on the diagonal line in the figure represent the squares, and the outer boxes represent the most-skewed rectangles with optimal perimeter. All integer points between (and including) the boxed points correspond to rectangles with minimum perimeter. Table 2 lists dimensions of the most skewed optimal rectangles corresponding to the boxed points above the diagonal.

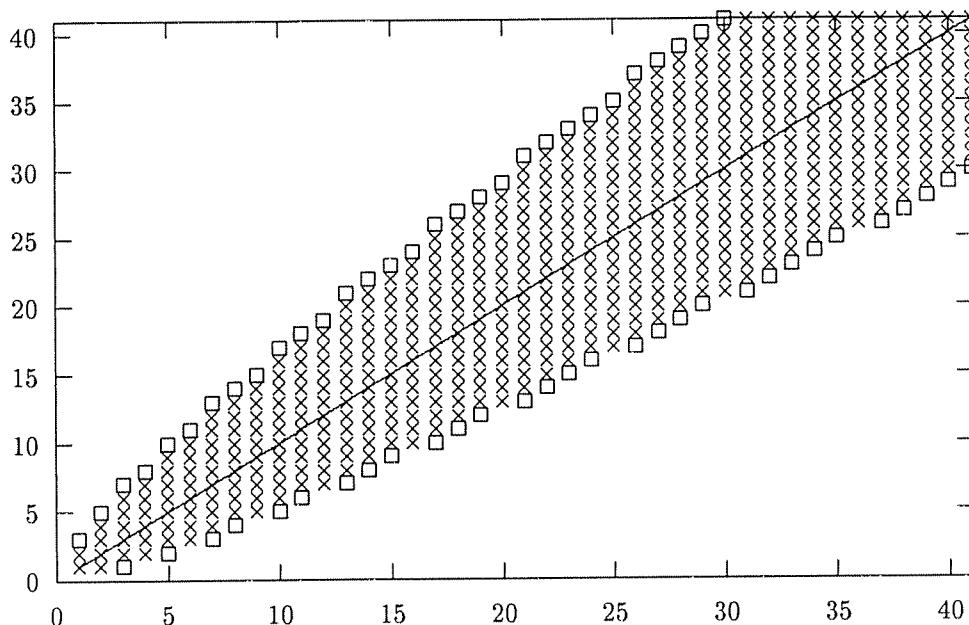


Figure 4: Dimensions of rectangles with optimal perimeter

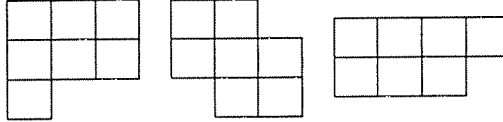
4.2 Optimal Irregular Configurations

The configuration occupied by a processor may be irregular (non-rectangular) and still be perimeter-optimal. The partial hypercubes introduced in section 3 are examples of such irregular configurations. In addition, some configurations that do not fall into the partial hypercube category are optimal. Consider a 2-dimensional case in which a processor's load is 7. Then the minimum achievable perimeter is $\mathcal{P}^*(7) = 6$.

k	Most-skewed perimeter-optimal rectangles $x \times (x + k)$				
2	1×3				
3	2×5				
4	3×7	4×8			
5	5×10	6×11			
6	7×13	8×14	9×15		
7	10×17	11×18	12×19		
8	13×21	14×22	15×23	16×24	
9	17×26	18×27	19×28	20×29	
10	21×31	22×32	23×33	24×34	25×35
11	26×37	27×38	28×39	29×40	30×41

Table 2: Some most-skewed perimeter-optimal rectangles

The following non-square configurations (and obvious variants) have perimeter six and are therefore perimeter-optimal:



The first configuration is a partial square constructed as described in section 3, but the others are not.

5 Optimal Tilings

In order to achieve the lower bound for θ_{total} , each processor must contribute exactly $\mathcal{P}^*(L_p)$ to the objective function. Thus, we wish to interleave perimeter-optimal configurations for all processors in order to fill the grid exactly. Note that since interchanging slices in the grid does not affect the objective function, the slices of a processor's configuration do not have to be contiguous, *i.e.*, the configuration for a processor need not be in compact form. In this section we exhibit classes of grids for which assignments which minimize various objective functions can be constructed by such tilings. Conditions under which assignments that minimize θ_{total} also minimize θ_{max} and other objective functions are also developed. These results generalize optimal assignments developed in Ghandeharizadeh *et al* [5].

5.1 Optimal Tilings for θ_{total}

One class of problems for which the minimum-perimeter results of section 3 yield optimal solutions are instances in which the grid can be tiled with hyper-rectangular blocks that are perimeter-optimal for each processor. In particular, if N can be factored as $f_1 f_2 \cdots f_D$ where f_i divides M_i ($i = 1, 2, \dots, D$) and the $\frac{M_1}{f_1} \times \frac{M_2}{f_2} \times \cdots \times \frac{M_D}{f_D}$

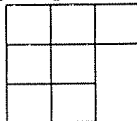
hyper-rectangular blocks are perimeter-optimal, then such a tiling is possible. Below we demonstrate an optimal assignment for such an instance: a 6×18 grid with 6 processors, each of which has a load of 18.

1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6

However, it is not necessary that all the blocks be oriented in the same way. An alternate optimal assignment for the same problem is shown below.

1	1	1	2	2	2	2	2	2	3	3	3	4	4	4	4	4	4
1	1	1	2	2	2	2	2	2	3	3	3	4	4	4	4	4	4
1	1	1	2	2	2	2	2	2	3	3	3	4	4	4	4	4	4
1	1	1	5	5	5	5	5	5	3	3	3	6	6	6	6	6	6
1	1	1	5	5	5	5	5	5	3	3	3	6	6	6	6	6	6
1	1	1	5	5	5	5	5	5	3	3	3	6	6	6	6	6	6

Another class of problems for which it is possible to construct assignments that are optimal for *both* θ_{total} and θ_{max} is the class of $N \times N$ grids with N processors, where each processor has a load of N . We have developed an algorithm that constructs optimal assignments for such problem instances [5]. Figure 5(a) shows the assignment produced by our algorithm when $N = 7$. Note that for any processor the slices in figure 5(a) may be permuted so that the set of cells occupied has the following optimal

configuration: . Figure 5(b) shows an optimal tiling of the same grid where the configurations are more compact.

5.2 Doubly Optimal Tilings for θ_{max} and θ_{total}

In this section we use a simple lower bound on θ_{max} to demonstrate several classes of rectangular grids for which both θ_{total} and θ_{max} may be simultaneously minimized.

Lemma 8 $\theta_{\text{max}} \geq \left\lceil \frac{\theta_{\text{total}}^*}{\sum_d M_d} \right\rceil$ where θ_{total}^* is the optimal value for θ_{total} .

Proof: The total of the ν 's is at least θ_{total}^* , so the average ν is at least $\frac{\theta_{\text{total}}^*}{\sum_d M_d}$. The maximum ν must be an integer at least as big as the average. \blacksquare

Besides the above $N \times N$ grid case, other tilings which are optimal for both θ_{max} and θ_{total} can be developed. In Ghandeharizadeh *et al* [5], we showed that

1	2	3	1	2	3	1
2	2	3	4	2	3	4
5	3	3	4	5	3	4
5	6	4	4	5	6	4
5	6	7	5	5	6	7
1	6	7	1	6	6	7
1	2	7	1	2	7	7

(a)

1	1	1	6	6	7	7
1	1	2	2	2	7	7
1	1	2	2	3	3	3
4	4	2	2	3	3	4
4	5	5	5	3	3	4
4	5	5	6	6	6	4
7	5	5	6	6	7	7

(b)

Figure 5: Some doubly optimal tilings

θ_{\max} achieves its lower bound for D dimensions if the grid can be tiled with hyper-rectangles whose proportions are the same as the proportions of the grid (*i.e.*, there are $N^{1/D}$ subdivisions along each dimension). If the corresponding hyper-rectangles are also perimeter-optimal, then such tilings are also optimal for θ_{total} by the results of section 3. In fact, lemma 8 shows that any tiling by perimeter-optimal configurations that results in the same chromatic index for each slice must be optimal for both θ_{\max} and θ_{total} . This observation also generalizes the $N \times N$ grid result of Ghandeharizadeh *et al* [5] (illustrated by Figure 5).

A *second* case in which both objective functions can be simultaneously minimized is when the grid can be tiled by perimeter-optimal configurations so that i slices have a chromatic index of k and the other j have an index of $k - 1$. In this case, $\theta_{\text{total}}^* = ki + (k - 1)j$, so the lower bound on θ_{\max} from lemma 8 is k . The tiling is therefore optimal for both objectives. The following grid is a two-dimensional example, in which the rows all have a chromatic index of three, the columns all have an index of two, and the grid is tiled by perimeter-optimal configurations.

1	1	1	2	2	2	3	3	3
4	4	4	5	5	5	6	6	6

A *third* doubly optimal case tiles the grid with perimeter-optimal configurations so that i slices have a chromatic index of k and the other j slices have an index of $k - 1$ or $k - 2$, with $i > j$. Again, $\theta_{\max} = k$ is optimal. To see this, note that the lower bound on θ_{\max} is at least $\left\lceil \frac{ki + (k-2)j}{i+j} \right\rceil = \left\lceil \frac{k(i+j) - 2j}{i+j} \right\rceil = \left\lceil k - \frac{2j}{i+j} \right\rceil = k$ since $\frac{2j}{i+j} < 1$. The following grid is a two-dimensional example of this third case with four processors assigned to each row and six assigned to each column.

1	1	1	2	2	2	3	3	3	4	4	4
5	5	5	6	6	6	7	7	7	8	8	8
9	9	9	10	10	10	11	11	11	12	12	12
13	13	13	14	14	14	15	15	15	16	16	16
17	17	17	18	18	18	19	19	19	20	20	20
21	21	21	22	22	22	23	23	23	24	24	24

5.3 Optimality for Other Objective Functions

The tilings that minimize the θ_{total} and θ_{max} functions also minimize more general objective functions. In order to verify the optimality of these assignments for other objective functions we make use of a relaxation of the original constraints of the problem. The original constraints were:

- (a) every cell of the grid is assigned to a processor
- (b) processor p has L_p cells assigned to it.

In section 3 we showed that constraint (b) implies

$$\sum_s \nu_s = \theta_{\text{total}} \geq \sum_p \mathcal{P}^*(L_p). \quad (6)$$

Therefore (6) is a relaxation of the original constraints, and for any objective function $f(\mathcal{V})$, where $\mathcal{V} = (\nu_1, \nu_2, \dots, \nu_{|S|})$, the continuous-variable problem

$$\begin{aligned} \min \quad & f(\mathcal{V}) \\ \text{s.t.} \quad & \sum_s \nu_s \geq \sum_p \mathcal{P}^*(L_p) \\ & \nu_s \geq 0 \quad \forall s \end{aligned} \quad (7)$$

is a relaxation of the integer-programming problem

$$\begin{aligned} \min \quad & f(\mathcal{V}) \\ \text{s.t.} \quad & \text{every cell is assigned to a processor,} \\ & \text{and each processor } p \text{ has } L_p \text{ cells assigned to it.} \end{aligned}$$

Consider the response time function $r(\nu) = \mathcal{O}^1 \nu + Q^1/\nu$ from section 1. In the absence of any constraints, the optimal number of processors per query (or equivalently, per slice) is the unconstrained minimum $\nu^* := (Q^1/\mathcal{O}^1)^{1/2}$. If the overhead $\mathcal{O}^1 N$ associated with communication with all of the N processors of the system is at least as great as the time Q^1 required to process the query on a single processor, then ν^* is at most $N^{1/2}$. Assume also that the (2-dimensional) grid given can be tiled with perimeter-optimal configurations so that there are exactly $N^{1/2}$ processors in each row and column (see, e.g., Fig. 5). (In hypercube architectures, it is not unusual to have N a perfect square.) Consider the continuous-variable problem

$$\begin{aligned} \min \sum_s r(\nu_s) = \min \sum_s (\mathcal{O}^1 \nu_s + Q^1/\nu_s) \\ \text{s.t.} \quad \sum_s \nu_s \geq \sum_p \mathcal{P}^*(L_p) \\ \nu_s \geq 0 \quad \forall s \end{aligned} \quad (8)$$

which is a relaxation of the form (7). In the tiling described above, $\nu_s^* = N^{1/2} \forall s$ and the constraint of (8) is satisfied as an equation. Since $\nu_s^* \geq \nu^*$, $r'(\nu_s^*) \geq 0$. Choosing

$r'(\nu_s^*)$ as a multiplier for the constraint of (8), optimality conditions for the convex problem (8) are satisfied. Thus, the tiling provides optimal processor assignments for all three objective functions θ_{total} , θ_{max} , and $\sum_s r(\nu_s)$. A similar argument applies to cases in higher dimensions in which $\nu_s^* = N^{\frac{D-1}{D}} > N^{1/2}$, and to arbitrary convex differentiable functions r with the property that $r'(\nu_s^*) \geq 0$.

6 Conclusions and Future Work

We have formalized the problem of partitioning data on a parallel database machine in order to minimize overhead. Lower bounds on the objective functions have been developed and we have demonstrated how the bounds can be attained in many cases by tiling with optimal configurations. To extend this work, we would like to describe exactly which hyper-rectangles have minimum D -perimeter for $D \geq 3$, and under which conditions tiling is possible. In addition, the square grid assignment algorithm may generalize to more than 2 dimensions. We would also like to explore other approaches to generating assignments for the data partitioning problem. A branch and bound type of approach in a suitably restricted search space seems promising. We also have a nonconvex nonlinear programming formulation of the problem that suggests other solution techniques. In addition, we would like to deal with more general objective functions and load balancing constraints as presented in Ghandeharizadeh *et al* [5]. Finally, it would be interesting to consider other applications that fit into the task assignment/parallel computing framework developed here (see, *e.g.*, Ghandeharizadeh *et al* [4]).

References

- [1] H. BORAL, W. ALEXANDER, L. CLAY, G. COPELAND, S. DANFORTH, M. FRANKLIN, B. HART, M. SMITH, AND P. VALDURIEZ, *Prototyping Bubba, a highly parallel database system*, IEEE Transactions on Knowledge and Data Engineering, 2 (1990), pp. 4–21.
- [2] D. DEWITT, S. GHANDEHARIZADEH, D. SCHNEIDER, A. BRICKER, H. HSIAO, AND R. RASMUSSEN, *The Gamma database machine project*, IEEE Transactions on Knowledge and Data Engineering, 2 (1990), pp. 44–63.
- [3] S. GHANDEHARIZADEH, *Physical Database Design in Multiprocessor Systems*, PhD thesis, Computer Sciences Department, University of Wisconsin - Madison, 1990.
- [4] S. GHANDEHARIZADEH, L. RAMOS, Z. ASAD, AND W. QURESHI, *Object placement in parallel hypermedia systems*, in Proceedings of the 1991 Very Large Data Bases Conference, Barcelona, Spain, September 1991.
- [5] S. GHANDEHARIZADEH, G. L. SCHULTZ, R. R. MEYER, AND J. YACKEL, *Optimal balanced assignments and a parallel database application*, Computer Sciences Technical Report 986, University of Wisconsin - Madison, Madison, WI, December 1990.

- [6] G. GRAEFE, *Volcano: An extensible and parallel dataflow query processing system*, computer science technical report, Oregon Graduate Center, Beaverton, OR, June 1989.
- [7] P. HELMAN, *A family of NP-complete data aggregation problems*, Acta Informatica, 26 (1989), pp. 485–499.
- [8] M. LIVNY, S. KHOSHAFIAN, AND H. BORAL, *Multi-disk management algorithms*, in Proceedings of the 1987 ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems, May 1987.
- [9] D. RIES AND R. EPSTEIN, *Evaluation of distribution criteria for distributed database systems*, UCB/ERL Technical Report M78/22, UC Berkeley, May 1987.
- [10] M. STONEBRAKER, D. PATTERSON, AND J. OUSTERHOUT, *The design of XPRS*, in Proceedings of the 1988 Very Large Data Bases Conference, Los Angeles, CA, September 1988.
- [11] TANDEM PERFORMANCE GROUP, *A benchmark non-stop SQL on the debit credit transaction*, in Proceedings of the 1988 SIGMOD Conference, Chicago, IL, June 1988.