Replication of Uniformly Accessed Shared Data for Large-Scale Data-Parallel Algorithms*

Chung-Ming Chen
Center for Biomedical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Soo-Young Lee
Department of Electrical Engineering
Auburn University
Auburn, AL 36849

Abstract

In this paper, we show how to minimize data sharing overhead required in most parallel algorithms, especially in Large-Scale Data-Parallel *(LSDP) algorithms, on a 2D mesh. Two specific issues are addressed in this study. One is what the optimal group size is, i.e., how many PEs should share a copy of shared data. The other is where the replicated data should be allocated.*

1 Introduction

Data sharing is inevitable in most parallel algorithms. For many problems, especially, for those problems with rich data parallelism, the overhead due to data sharing, i.e., data movement or data access contention [1], is one of the major factors degrading the performance of a parallel algorithm.

In this study, we investigate how to minimize the overhead caused by data sharing for a class of problems modeled as the *Large-Scale Data-Parallel* (LSDP) algorithms on a wrapped-around and a regular 2D meshes. An LSDP algorithm has the following features: (i) it has rich data-parallelism but without exclusive task and data partitioning, which means that some data need to be shared by multiple PEs, (ii) a great amount of shared data is involved (therefore, communication overhead is high), (iii) a synchronization point is required before any use of the shared data. This computation model may be found as the only or one of the major algorithmic structures in many applications, e.g., in an EM reconstruction algorithm for 3D Positron Emission Tomography [2].

Minimization of data sharing overhead for an LSDP algorithm is attempted by optimizing data access patterns in this study. A data access pattern specifies when and where to access shared data for each PE. Once all tasks have been assigned to PEs, the data sharing overhead is mainly determined by the data access pattern. In this paper, a uniform data access pattern is assumed, i.e., every PE has the same number of accesses to each of all other PEs. Our approach is to optimally replicate the shared data and allocate the replicated data. We also develop scheduling algorithms specifying shared data access sequences to

achieve the minimal (optimal) communication overhead.

Data replication is a widely used technique to enhance data locality at the expense of integrating and broadcasting replicated data. However, optimizing data replication has not been attempted in most previous works.

2 Models

2.1 System Model

The system topology to be considered in this paper is a wrapped-around and a regular (non-wrapped-around) 2D meshes with $N \times N$ PEs. Refer to [4] for the results on a hypercube.

A system is *bi-directional* if its links (i.e., the connections between adjacent PEs) are capable of realizing communication in both directions simultaneously. If each link of a system can perform communication in both directions but only one direction at a time, this system is considered as a *uni-directional* system. We assume that all links can perform communication independently.

In this study, analysis on a 2D mesh is based on the analysis on a linear array or a ring. For convenience, we use $WS_i, i = 1, 2$, to denote *uni-* and *bi-directional* wrapped-around 2D meshes, where the value of i indicates the number of directions realizable at a time by a link. Similarly, we use RS_i, $i = 1, 2$, to denote *uni-* and *bi-directional* regular 2D meshes, respectively.

Throughout this paper, we use the following convention to label PEs and links. For a ring or a linear array with m PEs, p_i denotes the ith PE, where $0 \le i \le (m-1)$ and l_i denotes the left link of p_i, where $0 \le i \le (m-1)$. For a wrapped-around or a regular 2D mesh with m rows and n columns of PEs, p_{ij} denotes the PE at the ith row and the jth column, hl_{ij} denotes the left (horizontal) link of p_{ij}, and vl_{ij} denotes the top (vertical) link of p_{ij}, where $0 \le i \le (m-1), 0 \le j \le (n-1)$. As an example, link labeling of a 4×3 ($m = 4$ and $n = 3$) wrapped-around 2D mesh is illustrated in Fig. 1, where the circle with a number ij inside represents p_{ij}.

2.2 Computation Model

Let M denote the total number of shared data, N_a the average number of accesses to each shared datum

for each PE and S_M the total number of shared data accesses by each PE which is equal to $N_a \times M$.

The computation model employed in this study, the *Large-Scale Data-Parallel* (LSDP) algorithm, consists of a large number of data-parallel computations. With multiple PEs, it is assumed that each PE performs a set of data-parallel computations beginning and ending with a synchronization point, respectively. Each data-parallel computation produces a *partial result* for a shared datum.

All PEs are divided into groups and all groups have the same number of PEs. Let n_g denote the group size, i.e., the number of PEs in a group. The shared data are replicated such that a copy of replicated data shared (which may be a subset of the entire shared data) by a group of PEs is evenly distributed among these PEs. All computations and associated shared data accesses are distributed over N_b pairs of computation and communication bands. Within each computation band, all PEs perform the same number of data-parallel computations.

To make the cost for each *computation band* independent of data replication for ease of analysis, all partial results computed by a PE are temporarily stored in the local memory of the PE. In the following communication band, each PE sends those partial results of which associated shared data are not in its local memory to the PEs which hold the associated shared data. Then, each PE *modifies* the shared data in its local memory using the partial results at the end of each communication band.

The overall time required for each communication band, defined as *updating time* and denoted by T_u, may be decomposed into two components. One component, denoted by T_{um}, is the time for all PEs to perform modifications using the partial results. The other component, denoted by T_{ux}, is the time required for all PEs to send the partial results to all other PEs in the same group.

If the shared data are replicated, before the synchronization point at the end, the (modified) replicated data in all PEs are *integrated* to ensure data coherence. Also, the integrated shared data are *broadcast* to all PEs. Note that the size of shared data involved in a broadcasting is n_g times larger than that in an integration.

The overall time required for integrating and broadcasting shared data is defined as *integration and broadcasting time*. Like the updating time, the integration and broadcasting time, denoted as T_I, is composed of two parts. One part, denoted as T_{Ix}, is the time for transmitting data and the other part, denoted as T_{Im}, is the time for performing the modifications on the replicated data. We use the integration and broadcasting algorithms proposed in [2]. The time required by the integration and broadcasting with N_I PEs is $\mathcal{O}((1 - 1/N_I)M)$ using the integration and broadcasting algorithms [2].

The goal of this study is to minimize the *data sharing* overhead, including *updating time* as well as *integration* and *broadcasting* time. It needs to be noted that *all time measurements used in the following discussions are normalized by the time to transfer one datum between two adjacent PEs during integration and broadcasting*. Let T_m denote the time required for one modification and T_x the time required for

transferring one datum between two adjacent PEs in a communication band.

3 Data Replication

In the following, we first discuss how the replicated data should be allocated. Then, the optimal group size is derived for the best type of allocation.

Due to the limited space, most of the propositions will be given without proofs. Refer to [4] for the proofs.

3.1 Two Types of Data Allocation

Two types of allocations may be considered for replicated data, namely, *aggregate* and *scatter*, as illustrated in Fig. 2 for a 2D mesh. If all PEs in each group, represented in the same pattern in Fig. 2 form a continuous region, it is called the *aggregate* type allocation. If each group is further divided into subgroups of the same size and the corresponding subgroups of all groups form a cluster as shown in Fig. 2, it is called the *scatter* type allocation.

For a 2D mesh, let the group size be $x \times y$ where $y \geq x$ and N is divisible by x and y. In this example (Fig. 2), PEs are divided into 9 groups indicated by 9 different patterns, each group with 4 PEs.

With a uniform data access pattern, the size of each packet of data, denoted as s, is equal to $N_a M/(N_b xy)$ on a 2D mesh.

We have performed a detailed formal comparison between the two allocations [4], which cannot be presented here due to the limited space. It has been shown *under our system and computation models* that the aggregate type allocation is not worse than the scatter type allocation.

3.2 T_{ux} and T_{Ix} with The *Aggregate* Type Allocation on A Mesh

To derive the optimal T_{ux} for RS_1, RS_2, WS_1 and WS_2 types of systems for the *aggregate* type allocation, we first compute the optimal T_{ux} for the 1D case and then extend the results to the 2D case. The general procedure to prove the optimality of T_{ux} and T_{Ix} for each system is to first find out the lower bound of the optimal T_{ux} and T_{Ix}, respectively, and then to show that these lower bounds are achievable by designing a scheduling algorithm.

3.2.1 Optimal T_{ux} for a linear array

Suppose that there are x PEs in a linear array. The optimal T_{ux} is derived in Proposition 3.1.

Proposition 3.1 *The optimal T_{ux} on a linear array is $\beta \lceil (x^2 - 1)/4 \rceil sT_x$, where β is 1 and 2 for a bi-directional and a uni-directional linear arrays, respectively.*

1D Updating Algorithm for a uniform DAP on a linear array

1. At step 0, p_0 sends different packets of data to all other PEs in the sequence of $p_{x-1}, p_{x-2}, ..., p_1$. The size of each packet is s. While the data

from p_0 to p_{x-1} are being sent, all other PEs except p_{x-1} also send a packet of data to p_{x-1} in parallel.

2. At step i, where $i < \lceil x/2 \rceil$, p_i sends a different packet of data to each remaining PE in the sequence of p_{x-1-i}, p_{x-2-i}, ..., p_{i+1}. While the data from p_i to p_{x-1-i} are being sent, all $p_k, i < k < (x-1-i)$, also send a packet of data to p_{x-1-i} in parallel.

This algorithm only describes the scheduling from left to right. Data accesses from right to left may be performed by using this algorithm but in a reversed direction.

3.2.2 Optimal T_{ux} for RS_1 and RS_2

On a regular 2D mesh with x rows and y columns of PEs and $y \geq x$, each datum is sent either horizontally followed by vertically or vertically followed by horizontally. Therefore, transferring data for updating on a regular 2D mesh is basically composed of two data transfers, each on linear arrays.

Proposition 3.2 *The optimal T_{ux} on a regular mesh with $x \times y$ PEs, is $\beta x \lceil (y^2 - 1)/4 \rceil sT_x$, where $y \geq x$ and β is 1 and 2 for an RS_2 and an RS_1 systems, respectively.*

2D Updating Algorithm for a uniform DAP on a regular 2D mesh

1. Divide each packet of data into two subpackets, namely, the *first-half* and the *second-half* subpackets, each with $s/2$ data.

2. Each row and column of PEs perform *1D Updating Algorithm for a uniform DAP on a linear array* twice for the corresponding linear arrays.

   - In the first phase, for all i and j, p_{ij} combines all first-half subpackets for all $p_{ak}, a = 0, ..., x-1$ and $k \neq j$, into a superpacket and sends it to p_{ik} along the ith row. Also, p_{ij} combines all second-half subpackets for all $p_{kb}, b = 0, ..., y-1$ and $k \neq i$, into a superpacket and sends it to p_{kj} along the jth column.

   - In the second phase, for all i and j, p_{ij} takes out the kth first-half subpackets of all superpackets received from the ith row, puts them in a new superpacket, and sends it to p_{kj} along the jth column. Also, p_{ij} takes out the kth second-half subpackets of all superpackets received from the jth column, puts them in a new superpacket, and sends it to p_{ik} along the ith row. The time required for shuffling subpackets is assumed to be negligible.

3.2.3 Optimal T_{ux} for a ring

As for a regular mesh, to derive the optimal T_{ux} for a wrapped-around mesh, we first derive that for a ring. Consider a ring with x PEs. For sending data from p_i

to p_j, define *left-distance* as the number of links in the left-path, $\{p_i, p_{(i-1+x)\bmod x}, ..., p_j\}$, and define *right-distance* as that in the right-path, $\{p_i, p_{(i+1)\bmod x}, ..., p_j\}$. To fully utilize the *wrapped-around* feature, each packet is sent through the path of min{*left-distance*, *right-distance*}. If the *left-distance* is equal to the *right-distance* for a packet, this packet is split into two subpackets and each of the *left-* and *right-paths* carries one subpacket. Therefore, the maximal distance between every send-receive pair of PEs is $\lfloor x/2 \rfloor$.

Proposition 3.3 *The optimal T_{ux} on a ring is $\beta \lceil (x^2 - 1)/4 \rceil sT_x/2$, where β is 1 and 2 for a bi-directional and a uni-directional rings, respectively.*

3.2.4 Optimal T_{ux} for WS_1 and WS_2

Like for a regular mesh, for a rectangular wrapped-around mesh with x rows and y columns of PEs and $y \geq x$, transferring data is also composed of two data transfers, each on rings.

Proposition 3.4 *The optimal T_{ux} on a wrapped-around mesh with $x \times y$ PEs, is $\beta x \lceil (y^2 - 1)/4 \rceil sT_x/2$, where $y \geq x$ and β is 1 and 2 for an WS_2 and an WS_1 systems, respectively.*

For comparison, in [3], the optimal T_{ux} has been shown to be $\Theta(p^{2/3})$ for a square wrapped-around mesh with $p \times p$ PEs.

Although a group (a submesh) in a wrapped-around 2D mesh is a regular mesh, the updating algorithm for a wrapped-around 2D mesh may be used by each group due to the *wrapped-around* feature of the whole mesh. Each submesh can actually be considered as a wrapped-around submesh by incorporating the rule: each PE in a group, without loss of generality, communicates with $\lfloor y/2 \rfloor$ PEs to its right and $y - \lfloor y/2 \rfloor - 1$ PEs to its left on its row as well as $\lfloor x/2 \rfloor$ PEs above it and $y - \lfloor x/2 \rfloor - 1$ PEs below it on its column. As a result, each submesh has the same optimal T_{ux} as if it were a wrapped-around submesh.

3.2.5 Optimal T_{Ix} for RS_1 and RS_2

For a regular 2D mesh, integration and broadcasting are done through a ring communication pattern. It is assumed that all corresponding PEs among groups on a regular 2D mesh can form a ring.

As an example, Fig. 3 illustrates one of the rings for integration. There are $N^2/(xy)$ PEs involved in the integration in each ring. It is easy to see that some links are shared by y rings, e.g., the central link on the first row in Fig. 3. It means that the data transferred in these y rings will pass through this central link during integration. The effective size of data passing through this central link in each ring is $(1 - xy/N^2)s$ [2], where s is the packet size which is $M/(2xy)$ for an RS_2 system and $M/(xy)$ for an RS_1 system. Therefore, the time required by the integration is $y(1 - xy/N^2)sT_x$. For broadcasting, all PEs are involved in the same broadcasting and the packet size is $M/2N^2$ and M/N^2 for an RS_2 and an RS_1 system, respectively. As a consequence, by using the Linear

integration and broadcasting algorithm, the optimal T_{Ix} normalized by M for a regular mesh, denoted as T_{Ix_r}, is

$$T_{Ix_r} = \frac{\beta}{2}(\frac{1}{x}(1 - \frac{xy}{N^2})) + \frac{\beta}{2}(1 - \frac{1}{N^2}) \qquad (1)$$

where $\beta = 1$ for an RS_2 system, $\beta = 2$ for an RS_1 system.

3.2.6 Optimal T_{Ix} for WS_1 and WS_2

For a wrapped-around 2D mesh, a ring pattern may be used for integration and broadcasting in each row and column. To fully utilize the *wrapped-around* characteristic of WS_2 and WS_1 systems, during integration, the data to be integrated are divided into two parts with sizes of $s_1 s$ and $s_2 s$ satisfying $s_1 + s_2 = 1$, where $s = M/(xy)$. While one part is integrated vertically followed by horizontally, the other part may be integrated horizontally followed by vertically. Similarly, during broadcasting, each packet is divided into two parts of the same size, i.e., $M/2N^2$. For a WS_2 system, each part may be further divided into two subparts of the same size, which are sent in two opposite directions of a ring. Then, normalized by M,

$$
\begin{aligned}
T_{Ix} = \; & \max\{\frac{\beta s_1}{2y}(1 - \frac{x}{N}), \frac{\beta s_2 y}{2x}(1 - \frac{y}{N})\} \\
& + \max\{\frac{\beta s_1}{2N}(1 - \frac{y}{N}), \frac{\beta s_2}{2N}(1 - \frac{x}{N})\} \\
& + \frac{\beta}{4}(1 - \frac{1}{N^2}).
\end{aligned}
$$

Proposition 3.5 T_{Ix} for a wrapped-around 2D mesh is minimized when $(s_1, s_2) = (s_{11}, s_{21})$, where

$$s_{11} = \frac{y(N - y)}{y(N - y) + x(N - x)}$$

$$s_{21} = \frac{x(N - x)}{y(N - y) + x(N - x)}$$

Therefore, the optimal T_{Ix} normalized by M for a wrapped-around mesh with a group size of $x \times y$, denoted as T_{Ix_w}, is

$$
\begin{aligned}
& \frac{\beta s_1}{2}(\frac{1}{y}(1 - \frac{x}{N}) + \frac{1}{N}(1 - \frac{y}{N})) + \frac{\beta}{4}(1 - \frac{1}{N^2}) \quad \text{if } S_1 \geq S_2 \\
& \frac{\beta s_2}{2}(\frac{1}{x}(1 - \frac{y}{N}) + \frac{1}{N}(1 - \frac{x}{N})) + \frac{\beta}{4}(1 - \frac{1}{N^2}) \quad \text{if } S_1 < S_2
\end{aligned}
$$
$$(2)$$
where $\beta = 1$ for a WS_2 system and $\beta = 2$ for a WS_1 system.

3.3 Optimal Data Replication on A Mesh

In this section, we would like to determine the optimal group size for data replication with a uniform data access pattern, using the aggregate type allocation.

Let T_r denote the overall data sharing time, normalized by M and the time for transferring one datum

between adjacent PEs, for a regular 2D mesh with a group size of $x \times y$. Recall that the overall data sharing time is $N_b(T_{um} + T_{ux}) + T_{Im} + T_{Ix}$, where T_{um}, T_{ux}, T_{Im} and T_{Ix} have been derived in the previous sections. Then,

$$T_r = \beta\lceil\frac{y^2 - 1}{4}\rceil\frac{N_a T_x}{y} + T_{Ix_r} + N_a T_m + (\frac{1}{xy} - \frac{1}{N^2})T_m$$

where $\beta = 1$ for an RS_2 system, $\beta = 2$ for an RS_1 system, and T_{Ix_r} as defined in Eq. (1). Similarly, let T_w denote the overall normalized data sharing time for a wrapped-around 2D mesh with a group size of $x \times y$,

$$T_w = \beta\lceil\frac{y^2 - 1}{8}\rceil\frac{N_a T_x}{y} + T_{Ix_w} + N_a T_m + (\frac{1}{xy} - \frac{1}{N^2})T_m$$

where $\beta = 1$ for a WS_2 system, $\beta = 2$ for a WS_1 system and T_{Ix_w} as defined in Eq. (2).

3.3.1 The Optimal Group Size on A Mesh

Proposition 3.6 *The optimal group size for a 2D mesh satisfies $x = y$.*

proof:
It can be shown that $\partial T_r/\partial x$ and $\partial T_w/\partial x$ are all negative. Therefore, the optimal group size should satisfy $x = y$ since $x \leq y$.

Q.E.D.

When $x = y$, all of T_r and T_w reduce to

$$
\begin{aligned}
T_o = \; & \alpha(\lceil\frac{x^2 - 1}{4}\rceil\frac{N_a T_x}{x} + \frac{1}{2}((\frac{1}{x} - \frac{x}{N^2}) + (1 - \frac{1}{N^2}))) \\
& + N_a T_m + (\frac{1}{x^2} - \frac{1}{N^2})T_m.
\end{aligned}
$$
$$(3)$$

where α, is 1, 2, 0.5 and 1 for an RS_2, an RS_1, a WS_2 and a WS_1 systems, respectively. Note that Eq. (3) is also applicable when $x = y = N$. The optimal group size, x_{opt}, for T_o may be determined by solving $\partial T_o/\partial x = 0$. Suppose that T_o is minimized at x_e and x_o when x is even and odd, respectively. Without giving a lengthy derivation, we have the following proposition.

Proposition 3.7 *Let $x_{eh} = \lceil x_e \rceil$, $x_{el} = \lfloor x_e \rfloor$, $x_{oh} = \lceil x_o \rceil$ and $x_{ol} = \lfloor x_o \rfloor$, where*

$$\Delta_{even} \geq 0 \; : \; x_e = (\frac{2N^2}{N^2 N_a T_x - 2})^{\frac{1}{3}}\left(\left(\frac{2T_m}{\alpha} + \Gamma_{even}\right)^{\frac{1}{3}} + \left(\frac{2T_m}{\alpha} - \Gamma_{even}\right)^{\frac{1}{3}}\right)$$

$$\Delta_{even} < 0 \; : \; x_e = 2$$

$$\Delta_{odd} \geq 0 \; : \; x_o = (\frac{2N^2}{N^2 N_a T_x - 2})^{\frac{1}{3}}\left(\left(\frac{2T_m}{\alpha} + \Gamma_{odd}\right)^{\frac{1}{3}} + \left(\frac{2T_m}{\alpha} - \Gamma_{odd}\right)^{\frac{1}{3}}\right)$$

$$\Delta_{odd} < 0 \; : \; x_o = 1$$

and

$$\Delta_{even} = 54N_aN^2T_xT_m{}^2 - \alpha^2N^2 - 108T_m{}^2$$

$$\Delta_{odd} = 432(N^2N_aT_x - 2)T_m{}^2 + \alpha^2N^2(N_aT_x - 2)^3.$$

$$\Gamma_{even} = \sqrt{\frac{4T_m{}^2}{\alpha^2} - \frac{2N^2}{27(N^2N_aT_x - 2)}}$$

$$\Gamma_{odd} = \sqrt{\frac{4T_m{}^2}{\alpha^2} - \frac{N^2(N_aT_x - 2)^3}{108(N^2N_aT_x - 2)}}$$

The optimal group size, x_{opt}, for data replication with a uniform data access pattern would be $x_{opt} \in \{x_{eh}, x_{el}, x_{oh}, x_{ol}\}$ for which T_o is minimal.

From Proposition 3.7, we can see that for a smaller α, the optimal group size tends to be larger. The reason is that as α decreases, T_{Im} becomes more influential in determining the optimal group size. Since T_{Im} decreases as the group size increases, a larger group size would be preferred for a smaller α.

4 Conclusions

In this paper, we have presented how to optimally allocate the replicated data and what the best group sizes are for the *Large-Scale Data-Parallel* algorithms on a mesh with a uniform data access pattern. Also, scheduling algorithms which specify data access sequences of PEs for the minimal data sharing time have been developed.

The significance of these results is that given an LSDP algorithm, one can easily determine the optimal data replication and allocation, and how to control the access sequence, so that the data sharing overhead is minimized when the data access pattern is uniform. Moreover, it may serve as a guideline for parallel compilers to determine the best data distribution, for example, in parallelizing a sparse matrix computation.

References

[1] M. Gupta and P. Banerjee, "Demonstration of automatic data partitioning techniques for parallelizing compilers on multicomputers," *IEEE Trans. Parallel Distributed Syst.*, vol. 3, pp. 179–193, Mar. 1992.

[2] C. M. Chen and S.-Y. Lee, "Parallelization of the EM algorithm for 3D PET image reconstruction: Performance estimation and analysis," in *Proc. 1991 Int. Conf. Parallel Processing*, vol. III, pp. 175–182, Aug. 1991.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Prentice-Hall, 1989.

[4] C. M. Chen, *On Miniminzing Data Sharing Overhead for Large-Scale Data-Parallel Algorithms: Replication and Allocation of Shared Data*. PhD thesis, Cornell University, Ithaca, New York, 1993.
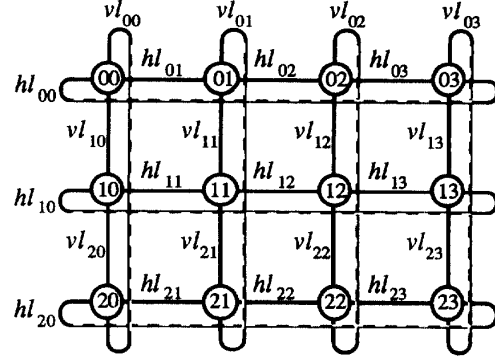
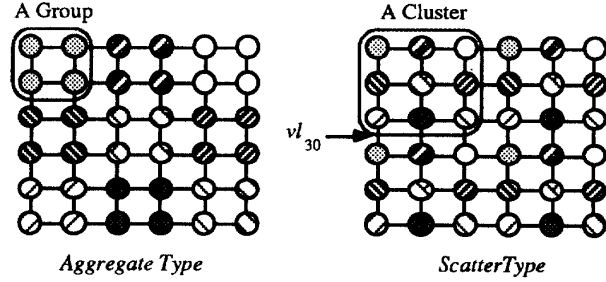Figure 1: An illustration of the System Model

Figure 2: Illustrations of the *aggregate* and the *scatter* type allocations on a mesh
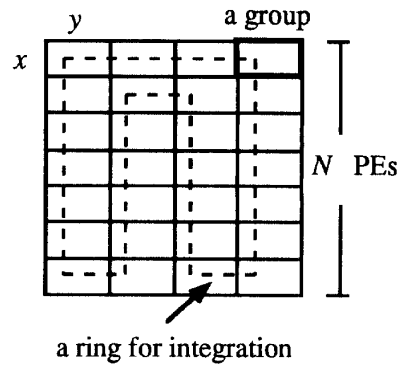

Figure 3: A ring for integration and broadcasting when N/y is even

250