CHAPTER III

PERMUTATION-COMPATIBLE DATA DISTRIBUTIONS

In this chapter, we discuss the data distribution of matrices for parallel matrix multiplication. First, we present the definition of *data distribution*. We also discuss two instances of data distributions (*i.e.*, the *block linear data distribution* and the *block scattered data distribution*) and give a general formula that can be used to derive these data distributions by choosing different parameters. Then, we define the notion of *permutation compatibility*, which is important for subsequently defining the *algorithmic compatibility* requirement for our parallel matrix multiplication algorithms. We discuss a permutation compatible data distribution (*i.e.*, the *virtual 2D torus wrap data distribution* (Huss-Lederman, Jacobson, and Tsao 1993; Huss-Lederman et al. 1994)) for distributing matrices on two-dimensional process grids. We also introduce a *modified virtual 2D data distribution* that can solve the potential load imbalance problem induced by the *virtual 2D torus wrap data distribution*. Finally, *algorithmic compatibility* is defined to ensure the correctness of our parallel dense matrix multiplication algorithms.

Data Distribution Functions

Data distribution is formally defined by van de Velde (1994) and a detailed definition of data distributions may be found in either (Skjellum 1990) or (Skjellum and Baldwin 1991).

Definition 1: (Data Distribution Functions) A data distribution function, μ, is a one-on-one mapping, $\mu(I, P, M) \mapsto (p, i)$, where I, $0 \leq I <$

M, is the global name of a coefficient, P is the number of processes among which all coefficients are to be partitioned; and M is the total number of coefficients. The pair (p, i) represents the process p ($0 \leq p < P$) and local (process-p) name i of the coefficient I. The inverse distribution function, $\mu^{-1}(p, i, P, M) \mapsto I$, transforms the local name (i), of process p back to the global coefficient name (I). The cardinality function, $\mu^{\sharp}(p, P, M) \mapsto m$, determines the number of coefficients (m) that belong to process p when M coefficients are distributed among P processes using data distribution function μ.

These mapping functions describe conversions between the global coefficient space and the local process and index space. Therefore, a specific data layout can be described by a set of data distribution functions. The data distribution functions are the basic abstractions needed to implement the data-distribution-independence (DDI) (van de Velde and Lorenz 1989; van de Velde 1990).

The *block linear data distribution* and the *block scattered data distribution* are two instances of data distribution. The partition of the global coefficients is performed in two steps. First, we partition the global coefficients into R blocks. Each block may or may not be of the same size, depending on whether R is a divisor of M. However, we try to partition M coefficients into blocks as the same size as possible. If we select the block size as $\lceil \frac{M}{R} \rceil$ or $\lfloor \frac{M}{R} \rfloor$, the blocks may vary at most one element. Furthermore, we choose to gather those blocks with slightly larger size in low numbered processes without loss of generality. The size of tth block is determined as follows:

$$b_t = \begin{cases} \lceil \frac{M}{R} \rceil & : \quad 0 \leq t < (M \bmod R) \\ \lfloor \frac{M}{R} \rfloor & : \quad (M \bmod R) \leq t < R. \end{cases} \tag{3.1}$$

Second, we assign blocks to processes consecutively. Successive blocks are assigned to different processes with a fixed stride s. The tth block is assigned to process p according to the following expression[2]:

$$p = \left(s \times \left(t \bmod P\right) + \left(\left(\frac{t}{\frac{P}{gcd(P,s)}}\right) \bmod gcd(P,s)\right)\right) \bmod P \qquad (3.2)$$

where s $(1 \leq s < P)$ is the stride of panel space and gcd denotes the greatest common divisor function.

By selecting different values of block number R and stride s in Equation 3.1 and 3.2, we can obtain various data distributions, some of which are useful. For example, if we select $R = P$ and $s = 1$, we will get the *linear data distribution*. If we select $R = M$ and $s = 1$, we will get the *scattered data distribution*. Furthermore, if we select other values of R and s, we will get *block scattered data distributions* with different block sizes and strides.

Data Distributions on Two-Dimensional Topologies

Now, we consider the distribution of a matrix on a two-dimensional process grid. A $P \times Q$ process grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$ is numbered as (p, q) where $p = 0 \ldots P-1, q = 0 \ldots Q-1$. Figure 3.1 shows a 4×3 process grid and the coordinate numbering scheme for each process.

When an $M \times N$ matrix A is mapped onto a process grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$, we perform the data mapping on both dimensions (*i.e.*, the row and column dimensions) independently with two instances of data distributions. In the row dimension, we denote the mapping function as $\mu(I, P, M) \mapsto (p, i)$. While, in the column dimension, we denote the mapping function as $\nu(J, Q, N) \mapsto (q, j)$. Figure 3.2 shows the data

[2] This expression is a modified version of the formula defined by *BiMMeR* (Huss-Lederman et al. 1994).

distribution of a matrix A on a grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$ with the *scattered data distribution* on the row dimension and the *linear data distribution* on the column dimension.

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)
(3,0)	(3,1)	(3,2)

Figure 3.1. Coordinate Numbering for a 4×3 Process Grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	$a_{4,6}$	$a_{4,7}$
$a_{8,0}$	$a_{8,1}$	$a_{8,2}$	$a_{8,3}$	$a_{8,4}$	$a_{8,5}$	$a_{8,6}$	$a_{8,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{5,0}$	$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	$a_{5,7}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$
$a_{6,0}$	$a_{6,1}$	$a_{6,2}$	$a_{6,3}$	$a_{6,4}$	$a_{6,5}$	$a_{6,6}$	$a_{6,7}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	$a_{3,6}$	$a_{3,7}$
$a_{7,0}$	$a_{7,1}$	$a_{7,2}$	$a_{7,3}$	$a_{7,4}$	$a_{7,5}$	$a_{7,6}$	$a_{7,7}$

Figure 3.2. The Data Distributions of a Matrix A on a Process Grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$

Permutation Compatibility for Matrix Multiplication

Permutation compatibility is the following property between two data distributions, which is also discussed in the early work by Falgout et al.(1992; 1993).

Definition 2: (Permutation Compatibility) Two distributions $\mu(\bullet, P, M)$ and $\nu(\bullet, Q, N)$ are permutation compatible if and only if $M = N$ and

$$\Lambda_{\mu}^{-1}(\mu(I, P, M), P, M) \quad = \quad \Lambda_{\nu}^{-1}(\nu(I, Q, M), Q, M)$$

$$\forall I \;=\; 0, \ldots, M-1$$

where the associated linear distributions Λ_μ and Λ_ν have matching cardinalities:

$$\begin{aligned} \Lambda_\mu^\sharp(p, P, M) &= \mu^\sharp(p, P, M) \\ \Lambda_\nu^\sharp(p, P, M) &= \nu^\sharp(p, P, M) \\ &\forall p = 0, \ldots, P-1. \end{aligned}$$

The concept of permutation compatibility is illustrated in Figure 3.3. Two different data distributions, $\mu(\bullet, 2, 6)$ and $\nu(\bullet, 3, 6)$, are permutation compatible, because the new orders of the global indices after distribution are identical.

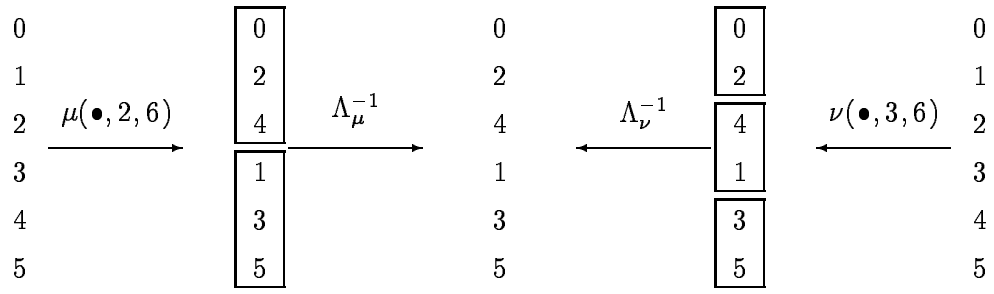

Figure 3.3. Illustration of the Concept of Permutation Compatibility

For the parallel matrix multiplication $C = \alpha AB + \beta C$, matrix A is defined to be an $M_A \times N_A$ matrix with a row distribution $\mu_A(\bullet, P, M_A)$ and a column distribution $\nu_A(\bullet, Q, N_A)$; and matrix B is defined to be an $M_B \times N_B$ matrix with a row distribution $\mu_B(\bullet, P, M_B)$ and a column distribution $\nu_B(\bullet, Q, N_B)$. Given these definitions, we can now discuss the permutation compatibility problem between ν_A and μ_B on a process grid $\mathcal{G}_{\mathcal{P} \times \mathcal{Q}}$.

1. If both ν_A and μ_B are *block linear data distributions*, ν_A and μ_B are permutation compatible on an arbitrary process grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$.

2. If both ν_A and μ_B are *block scattered data distributions*, in the case of a square process grid, ν_A and μ_B are permutation compatible; however, in the case of a non-square process grid, ν_A and μ_B are not permutation compatible. Note that the problem is caused by the different values of P and Q in Equation 3.2.

The permutation incompatibility of the *block scattered distribution* on non-square grids causes problem for the parallel matrix multiplication of $C = \alpha AB + \beta C$, because our algorithms requires the permutation compatibility between ν_A and μ_B. The permutation compatibility requirement is the global BLAS compatibility requirement (Falgout et al. 1992). If the column distribution of matrix A and the row distribution of matrix B satisfy the permutation compatibility requirement, we can utilize the optimal sequential assembly-coded version of *xGEMM* routines of BLAS (Dongarra et al. 1990) as local multiplication engine. Otherwise, we cannot use *xGEMM* routines. This can be resolved grossly by data remapping, but our point is not to resort to remapping within the classification scheme; we wish to avoid temporaries and excess communication as mentioned in (Bangalore 1995).

The Virtual Two-Dimensional Grid

For a non-square grid $\mathcal{G}_{\mathcal{P}\times\mathcal{Q}}$, we can view it as a $\alpha \times \alpha$ square virtual grid (Huss-Lederman, Jacobson, and Tsao 1993; Huss-Lederman et al. 1994), where α is the least common multiple of P and Q. Then we can distribute matrices on this $\alpha \times \alpha$ square virtual grid (Huss-Lederman et al. 1994). We modify the Equation 3.2 as follows to obtain the *virtual 2D torus wrap data distribution*:

$$p = \left(s \times \left(t \bmod \alpha\right) + \left(\left(\frac{t}{\frac{\alpha}{gcd(\alpha,s)}}\right) \bmod gcd(\alpha,s)\right)\right) \bmod \alpha \qquad (3.3)$$

If we choose the same block size and stride, ν_A and μ_B will prove to be permutation compatible. Furthermore, if we choose $s = \frac{\alpha}{P}$, we will obtain the same scattered data distribution as that obtained by Equation 3.2, but with local storage differences (process-local permutations) that are not important.

We achieve permutation compatibility between ν_A and μ_B by using the virtual grid, but we introduce a potential load imbalance problem. When N_A ($N_A = M_B$ for correctness) is not evenly divided by α, the remainder blocks are assigned to different virtual processes. However, these virtual processes may be in the same process, which causes more blocks on some processes than on other processes. Figure 3.4 illustrates the column distribution $\nu_A(\bullet, 6, 27)$ and the row distribution $\mu_B(\bullet, 4, 27)$ by using the *virtual 2D torus wrap data distribution*. For the column data distribution $\nu_A(\bullet, 6, 27)$, process 0 has one more element than the average. While for the row data distribution $\mu_B(\bullet, 4, 27)$, process 0 has two more elements than the average.

$q =$ Local Column Index

0	0	12	24	1	13	25
1	2	14	26	3	15	
2	4	16	5	17		
3	6	18	7	19		
4	8	20	9	21		
5	10	22	11	23		

$p =$ Local Row Index

0	0	12	24	1	13	25	2	14	26
1	3	15	4	16	5	17			
2	6	18	7	19	8	20			
3	9	21	10	22	11	23			

Figure 3.4. The Virtual 2D Torus Wrap Data Distribution

We can modify the distribution strategy to solve this problem and this is an important realization. After we use Equation 3.3, we actually obtain a new order for the global coefficients. We can then partition this new order of global coefficients into P parts so that each has $\lceil \frac{M}{P} \rceil$ or $\lfloor \frac{M}{P} \rfloor$ coefficients and consecutively assign them to the processes. We call this the *modified virtual 2D data distribution*. Figure 3.5

shows the column distribution $\nu_A(\bullet, 6, 27)$ and the row distribution $\mu_B(\bullet, 4, 27)$ using the *modified virtual 2D data distribution*, which achieves better load balance than the *virtual 2D torus wrap data distribution* does, shown in Figure 3.4.

$q =$	Local Column Index				
0	0	12	24	1	13
1	25	2	14	26	3
2	15	4	16	5	17
3	6	18	7	19	
4	8	20	9	21	
5	10	22	11	23	

$p =$	Local Row Index						
0	0	12	24	1	13	25	2
1	14	26	3	15	4	16	5
2	17	6	18	7	19	8	20
3	9	21	10	22	11	23	

Figure 3.5. The Modified Virtual 2D Data Distribution

Algorithmic Compatibility for Matrix Multiplication

Algorithmic compatibility is the property of *data distributions* of matrices A, B, and C that ensures an algorithm in the present study can be used correctly. The algorithms of matrix multiplication of the form $C = \alpha AB + \beta C$ require the following *algorithmic compatibility*, which is also shown in the early work by Falgout et al. (1992; 1993).

> Definition 3: (Algorithmic Compatibility) Matrices A, B, and C are compatible for the matrix multiplication algorithms if and only if the column distribution of A is permutation compatible with the row distribution of B; and both the row and column distributions of C are identical to the row distribution of A and the column distribution of B, respectively.

If we view the parallel matrix multiplication, $C = AB$, as $\mathcal{P}_C C \mathcal{Q}_C = (\mathcal{P}_A A \mathcal{Q}_A)(\mathcal{P}_B B \mathcal{Q}_B)$, where $\mathcal{P}_A$ (resp, $\mathcal{P}_B$ and $\mathcal{P}_C$) is a global permutation view

of A's (resp, B's and C's) row distribution, and $\mathcal{Q}_A$ (resp, $\mathcal{Q}_B$ and $\mathcal{Q}_C$) is a global permutation view of A's (resp, B's and C's) column distribution, then the compatibility definition above requires that $\mathcal{Q}_A = \mathcal{P}_B^T$. This means that

$$\mathcal{P}_C C \mathcal{Q}_C = (\mathcal{P}_A A \mathcal{Q}_A)(\mathcal{P}_B B \mathcal{Q}_B) = \mathcal{P}_A (AB) \mathcal{Q}_B = \mathcal{P}_A C \mathcal{Q}_B,$$

so that C has A's row distribution and B's column distribution.

A principal goal of the design of the parallel matrix multiplication algorithms presented in the study is data-distribution-independence (van de Velde and Lorenz 1989; van de Velde 1990). These algorithms do not require a specific data distribution, such as the *linear data distribution* or the *scattered data distribution*. When the data distributions of matrices A, B, and C satisfy the *algorithmic compatibility* requirement, these algorithms can be used correctly. Otherwise, prior redistribution of data is needed to ensure the correct result. The purpose of posing the restriction on the data distributions of matrices A, B, and C is to utilize x*GEMM* routines to achieve high performance. If performance issue were not considered, these algorithms could support all of the data distributions by using element-by-element multiplication instead of block-by-block multiplication, which would be extremely slow.