The Evaluation of Data Distributions for Multi-Dimensional Sparse Arrays Based on the EKMR Scheme

Chun-Yuan Lin+, Yeh-Ching Chung*2, and Jen-Shiuh Liu+

+Department of Information Engineering, Feng Chia University
Email: {cylin, liuj}@iecs.fcu.edu.tw

*Department of Computer Science, National Tsing-Hua University
Email: ychung@cs.nthu.edu.tw

Abstract

In our previous work, we have studied the performance of three data distribution schemes, Send Followed Compress (SFC), Compress Followed Send (CFS), and Encoding-Decoding (ED), for sparse arrays based on the traditional matrix representation (TMR) scheme. Since multi-dimensional arrays can also be represented by the extended Karnaugh map representation (EKMR) scheme, in this paper, we first apply the SFC, the CFS, and the ED schemes for multi-dimensional sparse arrays based on the EKMR scheme. Then, we compare the performance of these three schemes based on the EKMR scheme with those based on the TMR scheme. Both theoretical analysis and experimental test were conducted. In theoretical analysis, we analyze the SFC, the CFS, and the ED schemes based on the TMR and the EKMR schemes in terms of the data distribution time and the data compression time. In experimental test, we implemented these three schemes based on the TMR and the EKMR schemes on an IBM SP2 parallel machine. From the experimental results, the ED scheme outperforms the CFS scheme that outperforms the SFC scheme for most of test sparse arrays. The SFC, the CFS, and the ED schemes based on the EKMR scheme outperform those based on the TMR scheme, respectively.

Index Terms – Data distribution schemes, Data compression methods, Partition methods, Distributed memory multicomputers.

1. Introduction

Array operations are useful in a large number of important scientific codes, such as molecular dynamics [9], finite-element methods [15], climate modeling [28], etc. In the literature [2, 7, 17-18, 24-31, 33], many methods have been proposed to implement the data distribution schemes. These data distribution schemes are all classified to the Send Followed Compress (SFC) scheme for sparse arrays based on the traditional matrix representation (TMR) [19] scheme. A data distribution scheme for sparse arrays on a distributed memory multicomputer, in general, is composed of three phases, data partition, data distribution, and data compression. In the SFC scheme, these three phases are performed in the following order, the data partition, and then the data distribution, followed by the data compression.

Since parallel multi-dimensional array operations [7, 10, 16, 22-23] have been an extensively investigated problem, to propose efficient data distribution scheme for multi-dimensional sparse arrays become an important issue. In our previous work [20, 23], we have proposed two data distribution schemes, Compress Followed Send (CFS) and Encoding-Decoding (ED), for multi-dimensional sparse arrays based on the TMR scheme. In [20, 23], we have shown that the CFS and the ED schemes outperform the SFC scheme. The extended Karnaugh map representation (EKMR) [19] is another efficient array representation scheme for multi-dimensional arrays. Based on the EKMR scheme, in [21], we have proposed the EKMR-Compressed Row Storage (ECRS) and the EKMR-Compressed Column Storage (ECCS) data compression methods for multi-dimensional sparse arrays. In [21], we have shown that some multi-dimensional sparse array operations using the ECRS/ECCS methods outperform those using the Compressed Row Storage (CRS)/Compressed Column Storage (CCS) [5] methods based on the TMR scheme. Hence, in this paper, we first apply the SFC, the CFS, and the ED schemes for multi-dimensional sparse arrays based on the EKMR scheme. Then, we compare the performance of these three schemes based on the EKMR scheme with those based on the TMR scheme.

In order to evaluate these three schemes based on the EKMR scheme, in the data partition phase, the 2D mesh partition with load-balancing method is used. The details of the load-balancing method for 2D mesh partition can be found in [2, 30]. For the 2D mesh partition with load-balancing method, each

---

1 The work of this paper was partially supported by NSC under contract NSC91-2213-E-007-104.
2 The corresponding author.
processor has the same number of nonzero array elements. However, each processor has different size of local sparse array. In the data distribution phase, local sparse arrays, whether compressed or not, are sent to processors in sequence. In the data compression phase, the ECRS/ECCECS methods are used for the SFC and the CFS schemes. For the ED scheme, the encoding/decoding steps are used.

Bases on the methods used in the three phases, both theoretical analysis and experimental test were conducted. In theoretical analysis, we analyze the SFC, the CFS, and the ED schemes based on the TMR and the EKMR schemes in terms of the data distribution time and the data compression time. Here, we do not consider the data partition time since the comparisons of the data distribution time and the data compression time of these three schemes are based on the same partition methods. In experimental test, we implement these three schemes based on the TMR and the EKMR schemes on an IBM SP2 parallel machine. From the experimental results, based on the EKMR scheme, the ED scheme outperforms the CFS scheme that outperforms the SFC scheme. There are two reasons. First, we do not send entire local sparse arrays to processors in the CFS and the ED schemes. The data distribution time can be reduced. Second, for the ED scheme, the data distribution time is less than that of the CFS scheme. Besides, from the experimental results, the SFC, the CFS, and the ED schemes based on the EKMR scheme outperform those based on the TMR scheme, respectively. The reasons are two-fold. First, for the SFC scheme, the EKMR scheme can reduce the costs of packing non-continuous data blocks [22]. Second, for the SFC, the CFS and the ED schemes, the number of one-dimensional arrays used by the ECRS/ECCECS methods does not increase as the dimension increases. The time required to compress a sparse array can be reduced.

This paper is organized as follows. In Section 2, a brief survey of related work will be presented. We will briefly describe the EKMR scheme and the ECRS/ECCECS methods in Section 3. Section 4 will describe the SFC, the CFS, and the ED schemes for multi-dimensional sparse arrays based on the EKMR scheme in detail. Section 5 will analyze the theoretical performance of these three schemes based on the TMR and the EKMR schemes. The experimental results of these three schemes based on the TMR and the EKMR schemes will be given in Section 6.

2. Related Work

Many methods have been proposed to implement the data distribution schemes in the literature. Zapata et al. [2, 30] have proposed two data distribution schemes, Block Row Scatter (BRS) and the Multiple Recursive Decomposition (MRD). Based on the BRS and the MRD schemes, they [2-4, 7, 13-14, 26-30, 32] solve other important problems for sparse arrays. The BRS scheme is based on the division of any computation domain into several blocks, all of the same spatial shape and size. For the BRS scheme, each processor has the same size of local sparse array, yet each processor has different number of nonzero array elements. The MRD scheme can be considered as a generalization of the Binary Recursive Decomposition [6], a well-know data distribution scheme. For the MRD scheme, each processor has the same number of nonzero array elements, yet each processor has different size of local sparse array.

Ziantz et al. [33] proposed a run-time optimization technique that was applied to sparse arrays compressed by the CRS/CCS methods for array distribution and off-processor data fetching to reduce both the communication and computation time. They used the block data distribution scheme with a bin-packing algorithm.

Lee et al. [7] presented an efficient library for parallel sparse computations with Fortran 90 array intrinsic operations. They provided a data compression method, which is based on the CRS/CCS methods for two-dimensional sparse arrays, for multi-dimensional sparse arrays. Based on the MRD scheme, they also provide a data distribution scheme for multi-dimensional sparse arrays. Their approach is promising in speeding up sparse array computations using array intrinsic functions on both sequential and distributed memory environments.

3. Preliminary Concepts

Before presenting the SFC, the CFS, and the ED schemes for multi-dimensional sparse arrays based on the EKMR scheme, we briefly describe the EKMR scheme and the ECRS/ECCECS methods. Details of the EKMR scheme and the ECRS/ECCECS methods can be found in [19] and [21], respectively. In the following, we use TMR(n) and EKMR(n) for the TMR and the EKMR schemes of an n-dimensional array, respectively. In this paper, multi-dimensional sparse arrays based on the TMR and the EKMR schemes are all based on the row-major data layout [8].

3.1 The EKMR Scheme

Let A[i][j][l][k] denote a array based on the TMR(3) with a size of 3x4x5. The corresponding array A[4][15] based on the EKMR(3) is shown in Figure 1. In the EKMR(3), the index variable i is the same as the index variable i, whereas the index variable j is a combination of the index variables j and k. Figure 2 illustrates a corresponding array A[8][15] based on the EKMR(4) of array A[2][3][4][5]. In the EKMR(4), the index variable i is a combination of the index variables i and i. Based on the EKMR(4), we can generalize our results to the n-dimensional array.
3.2 The ECRS/IECCS Methods

The ECRS/IECCS methods use a set of three one-dimensional arrays, \( R \) and \( CK \), and \( V \), to compress a multi-dimensional sparse array based on the \( EKMR \) scheme. Given a sparse array based on the \( EKMR(3) \), the ECRS (IECCS) method compresses all of non-zero array elements along the rows (columns for IECCS) of the sparse array. Array \( R \) stores information of non-zero array elements of each row (column for IECCS). The number of non-zero array elements in the \( i \)th row (\( j \)th column for IECCS) can be obtained by subtracting the value of \( R[i] \) from \( R[i+1] \). Arrays, \( CK \) and \( V \), store the column (row for IECCS) indices and the values of non-zero array elements of each row (column for IECCS), respectively. The base of these three arrays is 0. For sparse arrays based on the \( EKMR(n) \), the ECRS/IECCS methods are similar to those of the \( EKMR(3) \). An example of the ECRS/IECCS methods for a sparse array based on the \( EKMR(3) \) is given in Figure 3.

4. The SFC, CFS and ED Schemes

In the following, we describe the SFC, the CFS, and the ED schemes for multi-dimensional sparse arrays based on the \( EKMR \) scheme in detail. We assume that a \( 4 \times 16 \) sparse array \( A \) based on the \( EKMR(3) \) with 16 non-zero array elements shown in Figure 4 and four processors are given. For the 2D mesh partition with load-balancing method, the four processors are treated as a \( 2 \times 2 \) processor array.

4.1 The SFC Scheme

In the SFC scheme, the data partition phase is performed first, then the data distribution phase, followed by the data compression phase. In the data partition phase, the partition result of the 2D mesh partition with load-balancing method \([2, 30]\) on \( A \) is shown in Figure 5. In the data distribution phase, local sparse arrays are sent to processors in sequence. Figure 6 shows the local sparse arrays received by each processor. In the data compression phase, the received local sparse array is compressed by the ECRS/IECCS methods. Figure 7 shows the compressed results by using the ECCS method. For the four- or higher dimensional sparse arrays based on the \( EKMR \) scheme, the SFC scheme is similar to that of the \( EKMR(3) \).
In the data distribution phase, arrays the partitioned arrays are compressed before send. In the data partition phase, the processes are the same as those of the SFC scheme. However, the values stored in array \( CK \) are global array indices because the partitioned arrays are compressed before send. In the data distribution phase, arrays \( R, CK, \) and \( V \) for each local sparse array are packed and sent to its corresponding processor. After received the corresponding packed buffer, each processor unpacks the buffer to the corresponding arrays \( R, CK, \) and \( V \). Since the values stored in array \( CK \) are global array indices, when unpack the received buffer, the values stored in array \( CK \) need to be converted to local array indices. We have the following cases.

Case 4.2.1. When the 2D mesh partition with load-balancing and the ECRS (ECCS) methods are used in the data partition and the data compression phases, respectively, each processor \( P_{ij} \) converts the values stored in array \( CK \) of the received buffer to the corresponding local array indices by subtracting \( M \) from each value stored in array \( CK \) of the received buffer. For \( P_{0,0}, P_{0,1}, \) and \( P_{1,1}, \) the packing, send/receive, and unpacking procedures are similar to that of \( P_{1,0} \). For the four- or higher dimensional sparse arrays based on the EKMR scheme, the CFS scheme is similar to that of the EKMR(3).

4.3 The ED Scheme

In the ED scheme, the data compression phase can be divided into two steps, encoding and decoding. In the ED scheme, the data partition phase is performed first, then the encoding step, followed by the data distribution phase and the decoding step. In the data partition phase, the process is the same as that of the SFC scheme. In the encoding step, each local sparse array based on the EKMR(3) is encoded into a special buffer \( B \). Figure 9 shows the formats of the special buffer \( B \) for sparse arrays based on the EKMR(3). In Figure 9, for the ECRS (ECCS) format, the \( R_i \) is used to store the number of nonzero array elements in a row (column for ECCS) \( i \). The \( C_{ij} \) and the \( V_{ij} \) are used to store the column (row for ECCS) index, and the value of the \( j \)th nonzero array element in a row (column for ECCS) \( i \), respectively. The \( C_{ij} \) and the \( V_{ij} \) are alternately stored in the buffer \( B \) and each \( C_{ij} \) is a global index of the global sparse array based on the EKMR(3).

![Figure 8](image1.png)  
(a) The data partition phase

![Figure 8](image2.png)  
(b) The data compression phase

![Figure 8](image3.png)  
(c) The data distribution phase

Figure 8: An example of the CFS scheme for a sparse array based on the EKMR(3).

| \( R \) | \( C_{K1} \) | \( V_{K1} \) | \( \ldots \) | \( C_{Kp} \) | \( V_{Kp} \) | \( \ldots \) | \( R_i \) | \( C_{ij} \) | \( V_{ij} \) | \( \ldots \) | \( R_k \) | \( C_{jk} \) | \( V_{jk} \) | \( \ldots \) | \( R_n \) | \( C_{nj} \) | \( V_{nj} \) | \( \ldots \) | \( R_m \) | \( C_{mj} \) | \( V_{mj} \) | \( \ldots \) |
| \( 1 \) | \( 2 \) | \( \ldots \) | \( K \) | \( \ldots \) | \( 1 \) | \( 2 \) | \( \ldots \) | \( i \) | \( \ldots \) | \( n \) | \( \ldots \) | \( 1 \) | \( 2 \) | \( \ldots \) | \( j \) | \( \ldots \) | \( m \) | \( \ldots \) | \( 1 \) | \( 2 \) | \( \ldots \) | \( k \) | \( \ldots \) | \( n \) |

(a) For the ECRS format

| \( R \) | \( V_{C1} \) | \( P_a \) | \( \ldots \) | \( V_{Cp} \) | \( P_a \) | \( \ldots \) | \( R_i \) | \( C_{ij} \) | \( V_{ij} \) | \( \ldots \) | \( R_k \) | \( C_{jk} \) | \( V_{jk} \) | \( \ldots \) | \( R_m \) | \( C_{mj} \) | \( V_{mj} \) | \( \ldots \) |
| \( 1 \) | \( 2 \) | \( \ldots \) | \( C \) | \( \ldots \) | \( 1 \) | \( 2 \) | \( \ldots \) | \( i \) | \( \ldots \) | \( n \) | \( \ldots \) | \( 1 \) | \( 2 \) | \( \ldots \) | \( j \) | \( \ldots \) | \( m \) | \( \ldots \) | \( 1 \) | \( 2 \) | \( \ldots \) | \( k \) | \( \ldots \) | \( n \) |

(b) For the ECCS format

Figure 9: The formats of the special buffer \( B \) for sparse arrays based on the EKMR(3).
In the data distribution phase, these special buffers are sent to processors in sequence. In the decodin
step, the special buffer B is decoded to get arrays R, CK, and V in each processor. To get array R, 
in each processor, R[0] is first initialized to 1. Then other values of array R are computed according to 
the formula $R[i+1] = R[i] + R[i]$, where $i = 0, 1, \ldots, n$ and $n$ is the number of rows in a local sparse array.
To get arrays CK and V, in each processor, we move all $C_{ij}$ and $V_{ij}$ stored in the special buffer to arrays CK and V, respectively, where $i = 0, 1, \ldots, n, j = 0, 1, \ldots, m$, $n$ is the number of rows of the local sparse array of a processor, and $m$ is the number of nonzero array elements in row $i$. Since each $C_{ij}$ is a global array index, to decode the received special buffer in the decoding step, each $C_{ij}$ need to be converted to a local array index. We have the following cases:

Case 4.3.1: When the 2D mesh partition with load-balancing method and the ECCS (ECCS) format
are used in the data partition phase and the encoding step, respectively, each processor $P_{i,j}$ converts each
$C_{i,j}$ of the received special buffer to the corresponding local array index by subtracting $M$
from each $C_{i,j}$ of the received special buffer, where $M$ is the total number of columns (rows for ECCS) in
$P_{i,0}, P_{i,1}, \ldots, P_{i,j-1}, (P_{0,j}, P_{1,j}, \ldots, P_{i,j})$ for ECCS.

An example of the ED scheme by applying the 2D mesh partition with load-balancing method and the
ECCS format on $A$ is given in Figure 10. Figure 10(a) shows the partition result. Figure 10(b) shows the special buffers for local sparse arrays. Figure 10(d) only shows the decoding step for $P_{i,0}$. After receiving the special buffer, to get array R, R[0] is first set to 1. Then other values of array R are computed according to the formula $R[i+1] = R[i] + R[i]$. To get array CK, we move $C_{0,0}$, $C_{5,0}$, and $C_{6,0}$ stored in the special buffer to array CK. According to Case 4.3.1 described above, $P_{i,0}$ subtracts 3 from $C_{0,0}$, $C_{5,0}$, and $C_{6,0}$ of the received special buffer to convert them to the desired local array indices. To get array V, we move $V_{0,0}$, $V_{5,0}$, and $V_{6,0}$ stored in the special buffer to array V. For $P_{0,0}, P_{0,1}$, and $P_{1,1}$, the decoding step is similar to that of $P_{i,0}$. For the four- or higher dimensional sparse arrays based on the EKMR scheme, the ED scheme is similar to that of the EKMR(3).

5. Theoretical Analysis

In this section, we analyze the SFC, the CFS, and the ED schemes based on the TMR and the
EKMR schemes in terms of the data distribution time and the data compression time. In the following, we
list the notations used in the theoretical analysis.

- $T_{Start}$ is the startup time for a communication channel.
- $T_{Data}$ is the transmission time for sending an array element through a communication channel.
- $T_{Operation}$ is the operation time for an array element. In order to simplify the analysis, we use $T_{Operation}$ to present any operation cost for an array element.
- $T_{Distribution}$ is the data distribution time for the data distribution phase. $T_{Distribution}$ includes the packing/unpacking time and send/receive time.
- $T_{Compression}$ is the data compression time for the data compression phase. For the ED scheme, $T_{Compression}$ is the sum of the encoding time and the decoding time in the encoding and the decoding steps, respectively.
- $A$ is a multi-dimensional sparse array based on the EKMR(3) format.
on the EKM R scheme.

- \( p \) is the number of processors. For the 2D mesh partition with load-balancing method, these \( p \) processors are treated as an \( r \times q \) processor array.
- \( s \) is the sparse ratio of a global sparse array.
- \( \alpha = \frac{\| A \|}{\| C \|} \) is set of space ratios of local sparse arrays. The space ratio for a local sparse array is the size of a local sparse array divided by the size of the global sparse array. The largest space ratio in \( \alpha \) is denoted as \( \alpha \).

Due to page limitation, first, we do not discuss these three schemes based on the EKM R scheme using the ECS method in this paper. The theoretical analysis results of these three schemes using the ECS method are similar to those of the ECRS method. Second, we do not discuss the analysis process of these three schemes based on the TMR scheme. The detail can be found in [23]. However, we will compare the theoretical and experimental performance of these three schemes for both the TMR and the EKM R schemes.

5.1 The SFC Scheme

Assume that an \( n \times n \) sparse array \( A \) is based on the EKM R(3) and \( p \) are given. The number of nonzero array elements in \( A \) is \( sn \). For the SFC scheme, the 2D mesh partition with load-balancing method partitions \( A \) into \( r \times q \) local sparse arrays and the number of nonzero array elements for each local sparse array is \( sn \times r \times q \). The largest local sparse array is \( r \times q = an^3 \). In the data distribution phase, local sparse arrays are sent to processors. For array \( A \) in the 2D mesh partition with load-balancing method, array elements in a local sparse array are not continuous. Therefore, local sparse arrays need to be packed before sending to processors in the data distribution phase. \( T_{\text{Data}} = \frac{r}{n} 	imes T_{\text{Startup}} + n \times T_{\text{Data}} + n \times T_{\text{Operation}} \). In the data compression phase, local sparse arrays are compressed by the ECS method using three arrays, \( R, CK \), and \( V \). \( T_{\text{Compression}} = \frac{n \times (n^2 \times (1+3r \times q))}{T_{\text{Operation}}} \).}

5.2 The CFS Scheme

For the CFS scheme, in the data compression phase, local sparse arrays are compressed by the ECS method using three arrays, \( R, CK \), and \( V \). \( T_{\text{Compression}} = \frac{n \times (n^2 \times (1+3r \times q))}{T_{\text{Operation}}} \). In the data distribution phase, the compressed results are first packed into buffers. These buffers are then sent to the corresponding processors. After receiving the corresponding buffer, each processor unpacks the buffer to get the desired arrays \( R, CK \), and \( V \). The values stored in array \( CK \) need to be converted to local sparse indices in each processor according to Case 4.2.1. \( T_{\text{Data}} = \frac{r}{n} \times T_{\text{Startup}} + (n^3 \times (3r \times q)) + T_{\text{Operation}} \).

5.3 The ED Scheme

For the ED scheme, in the encoding step, local sparse arrays are encoded into special buffers. In the data distribution phase, these special buffers are sent to processors. \( T_{\text{Data}} = \frac{r}{n} \times T_{\text{Startup}} + (2n^3 \times q) \times T_{\text{Data}} \). In the decoding step, the special buffer \( B \) in each processor is decoded. The \( C_{ij} \) stored in the special buffer needs to be converted to local sparse indices in each processor according Case 4.3.1. \( T_{\text{Compression}} = (\frac{n}{q} \times (1+\frac{3}{3}r \times q)) + T_{\text{Operation}} \).

Assume that an \( n \times k \) dimensional sparse array \( A \) is based on the EKM R scheme \( k > 3 \). The theoretical analysis method for array \( A \) is similar to that for a three-dimensional sparse array. Table 1 lists the data distribution time and the data compression time of the SFC, the CFS, and the ED schemes for an \( n \)-dimensional sparse array based on the TMR and the EKM R schemes.

5.4 Discussions

From Table 1, based on the EKM R scheme, we can see that \( T_{\text{Data}(ED)} \) is less than \( T_{\text{Data}(SFC)} \) and \( T_{\text{Data}(CFS)} \) if the sparse ratio \( s \) is less than 0.5. \( T_{\text{Data}(SFC)} \) is less than \( T_{\text{Data}(CFS)} \) if the sparse ratio \( s \) is less than 0.5 or the condition \( T_{\text{Data}} > (2s^2 + 2s)T_{\text{Operation}} \) is satisfied, respectively. If we assume that \( T_{\text{Data}} \) is equal to \( T_{\text{Operation}} \), \( T_{\text{Data}} > (2s^2 + 2s)T_{\text{Operation}} \) when \( s \) is less than 0.25. In [21], we have shown that the sparse ratio of an \( n \)-dimensional sparse array based on the EKM R scheme must be less than 0.5 if we want to use the ECRS/ECCS methods to compress it. Moreover, it shows that over 80% sparse array applications in which the sparse ratio of a sparse array is less than 0.1 according to the Harewell-Boeing Sparse Matrix Collection [11-12]. Therefore, for the data distribution time of these three schemes based on the EKM R scheme, we have two remarks.

**Remark 1.** \( T_{\text{Data}(ED)} \) is less than \( T_{\text{Data}(SFC)} \) and \( T_{\text{Data}(CFS)} \).

**Remark 2.** \( T_{\text{Data}(CFS)} \) is less than \( T_{\text{Data}(SFC)} \) for most of sparse arrays.

For the data compression time of these three schemes based on the EKM R scheme, we have the following remark.

**Remark 3.** \( T_{\text{Compression}(SFC)} \) is less than \( T_{\text{Compression}(CFS)} \) that is less than \( T_{\text{Compression}(ED)} \).

From Table 1, for the overall performance of these three schemes based on the EKM R scheme, we have two remarks.

**Remark 4.** The ED scheme outperforms the CFS scheme.

**Remark 5.** The ED and the CFS schemes outperform the SFC scheme if the conditions \( T_{\text{Data}} > (3s - \frac{3}{1} + 2s)r \times T_{\text{Operation}} \) and \( T_{\text{Data}} > (5s - \frac{5}{1} + 2s)r \times T_{\text{Operation}} \) are satisfied, respectively. \( (1/r \leq \alpha < 1) \)
Table 1: The data distribution time and the data compression time of these three schemes for an \( n \)-dimensional sparse array based on the TMR and the EKMR schemes.

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Scheme</th>
<th>Method</th>
<th>Complexity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TMR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SFC</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + n^3 x T_{\text{Data}} + n^3 \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (a + 4) r + q s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (3 n^3 s^q + q r) \times T_{\text{Data}} + (n^3 (3 + 4 r + q s) + r^2 + q n + q r + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 + 4 s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (3 n^3 s^q + q n r) \times T_{\text{Data}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 + (4 + 4 r + q s) s) + r^2 + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EKMR</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + n^3 \times T_{\text{Data}} + n^3 \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (a + 3) r + q s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td>( n = 3 )</td>
<td>CFS</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (2 n^3 s^q + q n r) \times T_{\text{Data}} + (n^3 (2 + 3 r + q s) + r^2 + q n + q r + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 + 3 s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td>( n = k ) (( k &gt; 3 ))</td>
<td>TMR</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (2 n^3 s^q + q n r) \times T_{\text{Data}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (a + k + 1) r s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EKMR</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (k n^3 s^q + q n r) \times T_{\text{Data}} + (n^3 k + (k + 1) r + s) + r^2 + q n + q r + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 k + s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (k n^3 s^q + q n r) \times T_{\text{Data}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 + (k + 1) r s) + s r + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SFC</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + n^3 \times T_{\text{Data}} + n^3 \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (a + 3) r + q s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (2 n^3 s^q + q n r) \times T_{\text{Data}} + (n^3 (2 + 3 r + q s) + r^2 + q n + q r + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 + 3 s) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>( T_{\text{Distribution}} )</td>
<td>( r x q \times T_{\text{Startup}} + (2 n^3 s^q + q n r) \times T_{\text{Data}} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_{\text{Compression}} )</td>
<td>( (n^2 \times (1 + 3 r + q s) + r^2 + n^2 + q r + 1) \times T_{\text{Operation}} )</td>
<td></td>
</tr>
</tbody>
</table>

From Table 1, for these three schemes based on the TMR and the EKMR schemes, we have three remarks.

Remark 6. The data distribution time of the SFC, the CFS, and the ED scheme based on the EKMR scheme is less than that based on the TMR scheme, respectively.

Remark 7. The data compression time of the SFC, the CFS, and the ED schemes based on the EKMR scheme is less than that based on the TMR scheme, respectively.

Remark 8. The SFC, the CFS, and the ED schemes based on the EKMR scheme outperform those based on the TMR scheme, respectively.

The reasons are two-fold. First, for the SFC scheme, the EKMR scheme can reduce the costs of packing non-continuous data blocks [22]. Second, for these three schemes, the number of one-dimensional arrays used by the ECRS/ECCS methods does not increase as the dimension increases. The time required to compress a sparse array can be reduced.

6. Experimental Results

In the experimental test, we implement the SFC, the CFS, and the ED schemes for multi-dimensional sparse arrays based on the TMR and the EKMR schemes on an IBM SP2 parallel machine. In the data partition phase, the 2D mesh partition with load-balancing method is implemented for these three schemes based on the TMR and the EKMR schemes, respectively. In the data compression phase, the CRS and the ECRS methods are implemented for sparse arrays based on the TMR and the EKMR schemes, respectively. All methods are written in C + MPI (Message Passing Interface) codes. The sparse ratio is set to 0.1 for all test three-dimensional sparse arrays.

Table 2 shows the data distribution time and the data compression time of these three schemes based on the EKMR scheme, we have the following observations.
Table 2: The data distribution time and the data compression time for three-dimensional sparse arrays.

<table>
<thead>
<tr>
<th>No. of Processors</th>
<th>Array Sizes-Schemes</th>
<th>$T_{\text{Distribution}}$</th>
<th>$T_{\text{Compression}}$</th>
<th>$T_{\text{Distribution}}$</th>
<th>$T_{\text{Compression}}$</th>
<th>$T_{\text{Distribution}}$</th>
<th>$T_{\text{Compression}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>60×60×60</td>
<td>120×120×120</td>
<td>180×180×180</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFC</td>
<td>T_{\text{Distribution}}</td>
<td>61.7</td>
<td>47.1</td>
<td>463.7</td>
<td>359.9</td>
<td>1756.3</td>
<td>1297.3</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>11.1</td>
<td>8.5</td>
<td>88.3</td>
<td>67.4</td>
<td>343.2</td>
<td>250.5</td>
</tr>
<tr>
<td>CFS</td>
<td>T_{\text{Distribution}}</td>
<td>14.7</td>
<td>9.4</td>
<td>121.3</td>
<td>84.3</td>
<td>444.1</td>
<td>281.6</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>46.2</td>
<td>31.1</td>
<td>380.3</td>
<td>252.8</td>
<td>1370.4</td>
<td>981.0</td>
</tr>
<tr>
<td>ED</td>
<td>T_{\text{Distribution}}</td>
<td>10.9</td>
<td>5.9</td>
<td>85.3</td>
<td>48.2</td>
<td>293.8</td>
<td>163.2</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>47.3</td>
<td>32.8</td>
<td>391.0</td>
<td>266.7</td>
<td>1474.4</td>
<td>1020.3</td>
</tr>
<tr>
<td>4x4</td>
<td>SFC</td>
<td>T_{\text{Distribution}}</td>
<td>72.6</td>
<td>48.3</td>
<td>504.1</td>
<td>353.0</td>
<td>2068.2</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>3.1</td>
<td>2.2</td>
<td>25.4</td>
<td>17.7</td>
<td>95.7</td>
<td>63.5</td>
</tr>
<tr>
<td>CFS</td>
<td>T_{\text{Distribution}}</td>
<td>21.5</td>
<td>14.2</td>
<td>134.1</td>
<td>95.7</td>
<td>492.9</td>
<td>383.4</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>40.7</td>
<td>32.5</td>
<td>387.3</td>
<td>265.2</td>
<td>1483.6</td>
<td>945.2</td>
</tr>
<tr>
<td>ED</td>
<td>T_{\text{Distribution}}</td>
<td>13.2</td>
<td>8.7</td>
<td>85.5</td>
<td>61.9</td>
<td>337.2</td>
<td>184.1</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>44.8</td>
<td>35.3</td>
<td>417.8</td>
<td>286.9</td>
<td>1575.5</td>
<td>1068.1</td>
</tr>
<tr>
<td>6x6</td>
<td>SFC</td>
<td>T_{\text{Distribution}}</td>
<td>88.2</td>
<td>55.1</td>
<td>568.8</td>
<td>369.9</td>
<td>2060.9</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>1.8</td>
<td>1.3</td>
<td>14.7</td>
<td>9.7</td>
<td>49.6</td>
<td>32.2</td>
</tr>
<tr>
<td>CFS</td>
<td>T_{\text{Distribution}}</td>
<td>18.3</td>
<td>10.8</td>
<td>127.0</td>
<td>101.2</td>
<td>456.9</td>
<td>300.1</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>66.5</td>
<td>37.6</td>
<td>445.5</td>
<td>267.2</td>
<td>1573.7</td>
<td>910.0</td>
</tr>
<tr>
<td>ED</td>
<td>T_{\text{Distribution}}</td>
<td>11.2</td>
<td>8.3</td>
<td>84.7</td>
<td>57.6</td>
<td>332.4</td>
<td>164.2</td>
</tr>
<tr>
<td></td>
<td>T_{\text{Compression}}</td>
<td>70.3</td>
<td>39.3</td>
<td>478.2</td>
<td>289.0</td>
<td>1657.2</td>
<td>1012.9</td>
</tr>
</tbody>
</table>

1. $T_{\text{Distribution}}(\text{ED}) < T_{\text{Distribution}}(\text{CFS}) < T_{\text{Distribution}}(\text{SFC}).$
2. $T_{\text{Compression}}(\text{ED}) > T_{\text{Compression}}(\text{CFS}) > T_{\text{Compression}}(\text{SFC}).$

These results match Remarks 1, 2 and 3. For the overall performance of these three schemes based on the EKMR scheme, from Table 2, we have the following observations. From experimental tests, we can estimate that $T_{\text{Data}} \approx 1.2 \times T_{\text{Operation}}$.

1. The CFS and the ED schemes outperform the SFC scheme since the conditions $T_{\text{Data}} > (\frac{5}{8}) T_{\text{Operation}}$ and $T_{\text{Data}} > (\frac{7}{8}) T_{\text{Operation}}$ shown in Table 1 are satisfied, respectively.
2. The ED scheme outperforms the CFS scheme.

These results match Remarks 4 and 5. From Table 2, for these three schemes based on the TMR and the EKMR schemes, we have the following observations.

1. The data distribution time of the SFC, the CFS, and the ED scheme based on the EKMR scheme is less than that based on the TMR scheme, respectively.
2. The data compression time of the SFC, the CFS, and the ED schemes based on the EKMR scheme is less than that based on the TMR scheme, respectively.
3. The SFC, the CFS, and the ED schemes based on the EKMR scheme outperform those based on the TMR scheme, respectively.

These results match Remarks 6, 7, and 8. From Table 2, we can see that the experimental results match the theoretical analysis shown in Table 1.

7. Conclusions

From the theoretical analysis and experimental results, for the SFC, the CFS, and the ED schemes based on the EKMR scheme, we have the following conclusions.

**Conclusion 1:** For the data distribution phase, $T_{\text{Distribution}}(\text{ED})$ is less than $T_{\text{Distribution}}(\text{SFC})$ and $T_{\text{Distribution}}(\text{CFS})$. For most of cases, $T_{\text{Distribution}}(\text{CFS})$ is less than $T_{\text{Distribution}}(\text{SFC})$.

**Conclusion 2:** For the data compression phase, $T_{\text{Compression}}(\text{SFC})$ is less than $T_{\text{Compression}}(\text{CFS})$ that is less than $T_{\text{Compression}}(\text{ED})$.

**Conclusion 3:** For the overall performance, the ED scheme outperforms the CFS scheme. For most of cases, the CFS and the ED schemes outperform the SFC scheme.

For these three schemes based on the TMR and the EKMR schemes, we have the following conclusions.

**Conclusion 4:** The data distribution time of the SFC, the CFS, and the ED scheme based on the EKMR scheme is less than that based on the TMR scheme, respectively.

**Conclusion 5:** The data compression time of the SFC, the CFS, and the ED schemes based on the EKMR scheme is less than that based on the TMR scheme, respectively.

**Conclusion 6:** The SFC, the CFS, and the ED schemes based on the EKMR scheme outperform those based on the TMR scheme, respectively.
References


