Contention-Free Communication Scheduling for Irregular Data Redistribution in Parallelizing Compilers

Kun-Ming Yu, Chi-Hsiu Chen, Ching-Hsien Hsu, Chang Wu Yu, and Chiu Kuo Liang

Department of Computer Science and Information Engineering, Chung Hua University, Hsinchu, Taiwan 300, ROC
Tel: 886-3-5186412, Fax: 886-3-5329701
yu@chu.edu.tw

Abstract. The data redistribution problems on multi-computers had been extensively studied. Irregular data redistribution has been paid attention recently since it can distribute different size of data segment of each processor to processors according to their own computation capability. High Performance Fortran Version 2 (HPF-2) provides GEN_BLOCK data distribution method for generating irregular data distribution. In this paper, we develop an efficient scheduling algorithm, Smallest Conflict Points Algorithm (SCPA), to schedule HPF2 irregular array redistribution. SCPA is a near optimal scheduling algorithm, which satisfies the minimal number of steps and minimal total messages size of steps for irregular data redistribution.

Keywords: Irregular data redistribution, communication scheduling, GEN_BLOCK, conflict points.

1 Introduction

More and more works had large data or complex computation on run-time in most scientific and engineering application. Those kinds of tasks require parallel programming on distributed system. Appropriate data distribution is critical for efficient execution of a data parallel program on a distributed computing environment. Therefore, an efficient data redistribution communication algorithm is needed to relocate the data among different processors. Data redistribution can be classified into two categories: the regular data redistribution [2, 3, 6] and the irregular data redistribution [1, 4, 10, 11, 12]. The irregular distribution uses user-defined functions to specify unevenly data distribution. High Performance Fortran version 2 (HPF2) provides GEN_BLOCK data distribution instruction which facilitates generalized unequal-size consecutive segments of array mapping onto consecutive processors. This makes it

* The work is partially supported by National Science Council of Taiwan, under grant number NSC-93-2213-E-216-029.
possible to let different processors dealing with appropriate data quantity according to their computation capability. In this scenario, all processors must send and receive message, even if send and receive on the same processor.

In the irregular array redistribution, Guo et al. [11] proposed a Divide-and-Conquer algorithm, they utilize Divide and Conquer technique to obtain near optimal scheduling while satisfied minimize the total communication messages size and minimize the number of steps.

In this paper, we present a smallest-conflict-points algorithm (SCPA) to efficiently perform GEN_BLOCK array redistribution. The main idea of the SCPA is to schedule the conflict messages with maximum degree in the first step of data redistribution process. SCPA can effectively reduce communication time in the process of data redistribution. SCPA is not only an optimal algorithm in the term of minimal number of steps, but also a near optimal algorithm satisfied the condition of minimal message size of total steps.

The rest of this paper is organized as follows. In Section 2, a brief survey of related work will be presented. In section 3, we will introduce communication model of irregular data redistribution and give an example of GEN_BLOCK array redistribution as preliminary. Section 4 presents smallest-conflict-points algorithm for irregular redistribution problem. The performance analysis and simulation results will be presented in section 5. Finally, the conclusions will be given in section 6.

2 Related Work

Many data redistribution results have been proposed in the literature. These researches are usually developed for regular or irregular problems [1] in multi-computer compiler techniques or runtime support techniques.

Techniques for communication optimizations category provide different approaches to reduce the communication overheads [5, 7] in a redistribution operation. The communication scheduling approaches [3, 12] avoid node contention and the strip mining approach [9] overlaps communication and computational overheads.

In irregular array redistribution problem, some works have concentrated on the indexing and message generation while some has addressed on the communication efficiency. Guo et al. [10, 11] proposed a divide-and-conquer algorithm for performing irregular array redistribution. In this method, communication messages are first divided into groups using Neighbor Message Set (NMS), messages have the same sender or receiver; the communication steps will be scheduled after those NMSs are merged according to the relationship of contention. Yook and Park [12] presented a relocation algorithm, while their algorithm may lead to high scheduling overheads and degrade the performance of a redistribution algorithm.

3 Preliminaries and Redistribution Communication Models

Data redistribution is a set of routines that transfer all the elements in a set of source processor S to a set of destination processor T. The sizes of the messages are specified
by values of user-defined random integer for array mapping from source processor to
destination processor. Since node contention considerably influences, a processor can
only send messages to other one processor in each communication step. Use the same
rule, a processor can only receive messages from other one processor.

To simplify the presentation, notations and terminologies used in this paper are
prior defined as follows.

**Definition 1**: GEN_BLOCK redistribution on one dimension array \(A[1:N]\) over \(P\)
processors. The source processor is denoted as \(SP_i\), the destination processor is
denoted as \(DP_j\), where \(0 \leq i, j \leq P-1\).

**Definition 2**: The time of redistribution separator the time of startup is denoted as \(t_s\),
and the time of communication is denoted as \(t_{\text{comm}}\).

**Definition 3**: To satisfy the condition of the minimum steps and the processor
sends/receives one message at each steps, some messages can not be scheduled in the
same communication step are called conflict tuple [11].

Data redistribution implements have two methods: non-blocking scheduling algo-
rithm and blocking scheduling algorithm. The non-blocking scheduling algorithm is
faster than the blocking scheduling algorithm. But need more buffer and be better
control synchronization. In this paper, we discuss on blocking scheduling algorithm.

Irregular data redistribution is unlike regular has a cyclic message passing pattern.
Every message transmission link is not overlapping. Hence, the total number of mes-
sage links \(N\) is \(numprocs \leq N \leq 2 \times numprocs - 1\), where \(numprocs\) is the num-
ber of processors. Figure 1 shows an example of redistributing two GEN_BLOCK
distributions on an array \(A[1:101]\). The communications between source and destina-
tion processor sets are depicted in Figure 2. There are totally fifteen communication
messages, \(m_1, m_2, m_3, \ldots, m_{15}\) among processors involved in the redistribution. In this
eample, \(\{m_2, m_3, m_4\}\) is a conflict tuple since they have common source processor
\(SP_1\); \(\{m_7, m_8, m_9\}\) is also a conflict point because of the common destination proces-
sor \(DP_4\). The maximum degree in the example is equal to 3. Figure 3 shows a simple
schedule for this example

<table>
<thead>
<tr>
<th>Source distribution</th>
<th>Source Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(SP_0)</td>
</tr>
<tr>
<td>Size</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Size</td>
</tr>
</tbody>
</table>

**Fig. 1.** An example of distributions
3.1 Explicit Conflict Point and Implicit Conflict Point

The total communication time of a message passing operation using two parameters: the startup time $t_s$ and the unit data transmission time $t_m$. The startup time is once for each communication event and is independent of the message size to be communicated. The data transmission time is a relationship of a message size, $\text{size}(m)$. The communication time of one communication step is the maximum of the message in this step. The total communication time of all steps is the sum of each communication time of step. The length of these steps determines the data transmission overheads. The minimum step is equal to the maximum degree $k$, when a message cannot be put into any step of minimum step it must relate to the processor has maximum degree transmission links. Figure 4 shows the maximum degree of figure 1. SP$_1$, SP$_2$ and DP$_4$ had maximum degree ($K = 3$) from messages m$_2$~m$_9$. Because of each one processor can only send/receive at most one message to/from other processor in each communication step. First, we concentrate all processors which have maximum degree transmission links messages. For the sake of simplicity, such messages are referred to as “Maximum Degree Message Set” (MDMS) in the paper, as shown in figure 4. If the messages in MDMSs can put into $k$ steps with no conflict occur, other messages of the processors’ degree less than maximum degree will be easier to put into the rest of step without increasing the number of steps.

We say a message to be an explicit conflict point if it belongs to two MDMSs. There exists at most one explicit conflict point between two MDMSs. In figure 4, m$_7$ is a explicit conflict point since it belongs to two MDMSs \{m$_5$, m$_6$, m$_7$\} and \{m$_7$, m$_8$, m$_9$\}. On the other hand, if two MDMSs do not contain the same message, but the
neighbor MDMSs each has a message been sent by the same processor, or been received by the same processor. We call this kind of message as an implicit conflict point. As shown by figure 5, m₄ and m₅ are contained by the different MDMSs. DP₂ only receives m₄ and m₅ two messages, so it can not form an MDMS. But m₄ and m₅ are also owned by different MDMSs. Therefore, m₄ is an implicit conflict point. Although, m₅ is also covered by two MDMSs, but it is restricted by m₄. Hence m₅ will not cause conflict. Figure 7 depicts all MDMSs for the example shown in Figure 1.

![Fig. 4. Maximum Degree Messages Set](image1)

![Fig. 5. Example of explicit conflict point](image2)

### 4 Scheduling Algorithm

The main goal of irregular array distribution is to minimize communication step as well as the total message size of steps. We select the smallest conflict points which will really cause conflict to loose the schedule constraint and to minimize the total message size of schedule.

Smallest conflict points algorithm consists of four parts:

1. Pick out MDMSs from given data redistributed problem.
2. Find out explicit conflict point and implicit conflict point. And schedule all the conflict point into the same schedule step.
3. Select messages on MDMSs in non-increasing order of message size. Schedule message into similar message size of that step and keep the relation of each processor send/receive at most one message to/from the processor. Repeat above process until no MDMSs’ messages left.
4. Schedule messages do not belong to MDMSs by non-increasing order of message size. Repeat above process until no messages left.

From Figure 1, we can pick out four MDMSs, MDMS₁ = {m₂, m₃, m₄}, MDMS₂ = {m₄, m₅}, MDMS₃ = {m₅, m₆, m₇} and MDMS₄ = {m₇, m₈, m₉}, shown in Figure 8. We schedule m₄ and m₇ into the same step. Then schedule those messages on
MDMSs by non-increasing order of message size as follows: \( m_8, m_3, m_5, m_6, m_2, m_9 \). After that, we can schedule the rest messages that are not belong to any MDMSs by non-increasing order of message size as follows: \( m_1, m_{15}, m_{10}, m_{12}, m_{13}, m_{14}, m_{11} \).

Figure 9 shows the final schedule obtained from smallest conflict points algorithm.

5 Performance Evaluation and Analysis

To evaluate the performance of the proposed methods, we have implemented the SCPA along with the divide-and-conquer algorithm [11]. The performance simula
tion is discussed in two classes, even GEN_BLOCK and uneven GEN_BLOCK distributions. In even GEN_BLOCK distribution, each processor owns similar size of data. Contrast to even distribution, few processors might be allocated grand volume of data in uneven distribution. Since array elements could be centralized to some specific processors, it is also possible for those processors to have the maximum degree of communications.

The simulation program generates a set of random integer number as the size of message. To correctly evaluate the performance of these two algorithms, both programs were written in the single program multiple data (SPMD) programming paradigm with MPI code and executed on an SMP/Linux cluster consisted of 24 SMP nodes. In the figures, “SCPA Better” represents the percentage of the number of

![Graph](image1)

**Fig. 10.** The events percentage of computing time is plotted (a) with different number of processors and (b) with different of total messages size in 8 processors, on uneven data set
events that the SCPA has lower total steps of messages size than the divide-and-conquer algorithm (DCA), while “DCA Better” gives the reverse situation. In the uneven distribution, the size of message’s up-bound is set to \((\text{totalsize}/\text{numprocs})\times1.5\) and low-bound is set to \((\text{totalsize}/\text{numprocs})\times0.3\), where totalsize is total size of messages and numprocs is the size of processor. In the even distribution, the size of message’s up-bound is set to \((\text{totalsize}/\text{numprocs})\times1.3\) and low-bound is set to low-bound \((\text{totalsize}/\text{numprocs})\times0.7\). The total messages size is 1M.

Figure 10 shows the simulation results of both the SCPA and the DCA with different number of processors and total message size. We can observe that SCPA has better performance on uneven data redistribution compared with DCA.

Since the data is concentrated in the even case, from figure 11, we can observe that SCPA have the better performance compared with uneven case. Figure 11 also

![Graph](image1)

(a)

![Graph](image2)

(b)

Fig. 11. The events percentage of computing time is plotted (a) with different number of processors and (b) with different of total messages size in 8 processors, on even data set.
illustrates that SCPA has at least 85% supreme than DCA in any size of total messages and any number of processors. In both even and uneven case, SCPA performs slightly better than DCA.

6 Conclusion

In this paper, we have presented an efficient scheduling algorithm, smallest conflict points algorithm (SCPA), for irregular data distribution. The algorithm can effectively reduce communication time in the process of data redistribution. Smallest-conflict-points algorithm is not only an optimal algorithm in the term of minimal number of steps, but also a near optimal algorithm satisfied the condition of minimal message size of total steps. Effectiveness of the proposed methods not only avoids node contention but also shortens the overall communication length.

For verifying the performance of our proposed algorithm, we have implemented SCPA as well as the divide-and-conquer redistribution algorithm. The experimental results show improvement of communication costs and high practicability on different processor hierarchy. Also, the experimental results indicate that both of them have good performance on GEN_BLOCK redistribution. But also both have advantages and disadvantages. In many situations, SCPA has better than the divide-and-conquer redistribution algorithm.

References


