A Framework for Exploiting Data Availability to Optimize Communication

Manish Gupta and Edith Schonberg
IBM T. J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Abstract. This paper presents a global analysis framework for determining the availability of data on a virtual processor grid. The data availability information obtained is useful for optimizing communication when generating SPMD programs for distributed address-space multiprocessors. We introduce a new kind of array section descriptor, called an Available Section Descriptor, which represents the mapping of an array section onto the processor grid. We present an array data-flow analysis procedure, based on interval analysis, for determining data availability at each statement. Several communication optimizations, including redundant communication elimination, are also described. An advantage of our approach is that it is independent of actual data partitioning and representation of explicit communication.

1 Introduction

Distributed memory architectures are becoming increasingly popular as a viable and cost-effective method of building massively parallel computers. However, the absence of global address space, and consequently, the need for explicit message passing among processes makes such machines very difficult to program. This has motivated considerable research towards developing compilers that relieve the programmer of the burden of generating communication [15, 23, 17, 19, 18, 16, 21]. Such compilers take a sequential or a shared-memory parallel program, annotated with directives specifying data decomposition, and generate the target SPMD program with explicit message passing. Thus, the compiler performs two essential tasks:

- partitioning of computation, usually based on the owner computes rule [15, 23, 19], which makes the processor that owns a data item (being assigned a value) responsible for its computation, and
- generation of communication, whereby the owner of a data item (being used in a computation) sends its value to the processor performing that computation.

In order to reduce the cost of communication, the compilers perform optimizations like message vectorization [15, 23], using collective communication [13, 17], and overlapping communication with computation [15]. However, the extent to
which most compilers are able to (or even attempt to) optimize communication is still limited by the following factors:

- It is always the owner of a data item that supplies its value when required by another processor. It may be possible to generate more efficient communication if any processor that has a valid copy of that data is allowed to send its value. In the special case when a processor receiving a value via communication already has a valid copy of that data item available, the communication can be recognized as being redundant, and eliminated.

- There is no global analysis performed of the communication requirements of array references over different loop nests. This precludes general optimizations, such as redundant communication elimination, or carrying out extra communication inside one loop nest if it subsumes communication required in the next loop nest.

As an example, consider the program fragment in Figure 1. According to the HPF alignment directives, B and A are identically aligned to an abstract template, represented by VPROCS. The variable D is aligned with the first column of VPROCS. To enforce the owner computes rule at statement $s_3$, communication must be generated which creates a copy of A transpose aligned with B. Similarly, statement $s_6$ would require a second communication to align the first row of A with D. However, since D is aligned with the first column of B, the second communication is subsumed by the the first, and is therefore redundant.

Our goal is to provide a framework, based on global array data-flow analysis, for performing communication optimizations. The conventional approach to data-flow analysis regards each access to an array element as an access to the entire array. Previous researchers [12, 11, 20] have applied data-flow analysis to array sections to improve its precision. The array section descriptors, such as the regular section descriptor (RSD) [7], and the data access descriptor (DAD) [5], provide a compact representation for elements comprising certain special substructures of arrays. However, in the context of communication optimizations,