Chapter 11

Introduction to the C* Communication Library

Chapters 11–14 of this guide describe a set of C* library functions that provide different kinds of communication. For example, these functions allow you to:

- Send values of parallel variable elements to other elements of the same shape.
- Send values of parallel variable elements of one shape to elements of another shape.
- Perform different kinds of computation on values while sending them to elements of the same or a different shape.
- Send data from parallel variable elements to a scalar variable, and from a scalar variable to a parallel variable element.
- Send data from a parallel variable to a scalar array, or from an array to a parallel variable.

Of course, you can perform similar kinds of communication using features of C* itself; see Chapter 10. These library functions supplement, and in many cases overlap, the communication features contained in the language itself. Several of them are particularly useful when the rank of a shape is not known until run time; in that situation, you cannot use left indexing to specify a parallel variable element, because you cannot specify values for all the axes when you write the program. The functions, however, provide a way to manipulate such data.

This chapter introduces the methods of communication available using C* library functions, and gives an overview of these functions.

Include the header file **<cscomm.h>** in programs that call any of the functions discussed in the next three chapters. The functions are part of the C* run-time system, and are linked in to your program by default.

11.1 Two Kinds of Communication

There are two different kinds of communication in C*: *grid* and *general*.

11.1.1 Grid Communication

In grid communication, elements of parallel variables in the same shape communicate in regular patterns by using their coordinates. In other words, values of all elements in a parallel variable move the same number of positions in the same direction — for example, each element sends its value to the element of another parallel variable that is two coordinates higher along axis 0.

These functions implement grid communication:

- **from_grid**
- **from_grid_dim**
- **from_torus**
- **from_torus_dim**
- **to_grid**
- **to_grid_dim**
- **to_torus**
- **to_torus_dim**

In addition, the **pcoord** function, which we discussed in Chapter 10, can be used in certain kinds of grid communication.

Grid communication is discussed in Chapter 12.

11.1.2 General Communication

General communication allows any parallel variable element to send its value to any other element, whether or not they are of the same shape, and whether or not the pattern of communication is regular. It also allows scalar variables to send values to or receive values from parallel variables. This kind of communication uses a position's *send address* rather than its coordinates. The send address is a combination of a position's shape and coordinates that uniquely identifies the position among all positions in all shapes. General communication is more versatile than grid communication, but it is also slower. It achieves the same result as parallel left-indexing a parallel variable; see Chapter 10.

General communication is implemented by these C* functions:

- **`make_send_address`**
- **`send`**
- **`get`**
- **`read_from_position`**
- **`read_from_pvar`**
- **`write_to_position`**
- **`write_to_pvar`**
- **`make_multi_coord`**

These functions are discussed in Chapter 14.

11.2 Communication and Computation

Many C* functions perform computations or combining operations on the parallel values they transmit. Most of these functions involve grid communication. For example, the **`scan`** function lets you combine values of specified elements of a parallel variable along an axis of a shape. You can add these values, for example, multiply them, or take the minimum or maximum. These C* library functions provide communication and computation:

- **`scan`**
- **`spread`**

- **copy_spread**
- **multispread**
- **copy_multispread**
- **enumerate**
- **rank**
- **reduce**
- **copy_reduce**
- **global**

These functions are discussed in Chapter 13.