1 Schematic view of the remap implementation

The `remap` operation copies an array `src` to an array `dst`. The two arrays must have the same shape (and type) but their mappings need not be related.

The Adlib implementation is considered to be essentially "optimal" in the sense that that it minimizes the number of block communication operations required to implement the data redistribution. In this context a "block communication operation" is a send or receive operation on a regular `R`-dimensional block of memory, parametrized by a base address, `R` extents, and `R` memory strides. Because arbitrary strides are allowed, a block is not necessarily contiguous in memory: MPI allows sending or receiving such non-contiguous blocks through derived-type constructors such as `MPI_Type_hvector`.

The `remap` operation is implemented in two stages:

1. construct a *communication schedule*, then

2. execute that schedule.

A communication schedule can be implemented by three lists in each processor: a list of block send operations, list of block receive operations, and a list of local block memory copy operations.

The scheme for constructing the "send" component of the communication schedule is illustrated in figures 1, 2 and 3. In figure 1 we see that the source array is mapped into a processor grid with some blocks on each processor (in the picture `src` has a block cyclic distribution in one of its dimensions). On each processor, we enumerate the locally held blocks in an outer loop. One such block is highlighted. This block represents some regular section of the `src` array. This section has to be copied to the corresponding regular section of the `dst` array. Figure 2 illustrates how the latter regular section might be mapped to the processor grid. We enumerate the blocks of the target section: in this case redistributing the highlighted block requires one local copy operation and three send operations, as illustrated in figure 3.

An analoguous procedure constructs the receive schedule.

As block sends and receives are enumerated they are placed in queues, prioritized by the difference between source and destination process ids (modulo the number of processes). Messages of "hop" 1 will be sent first, followed by messages of hop 2, and so on (hop 0 corresponds to local memory copies). During the "execute" phase of the `remap` sends and receives occur in different logical threads (which are simulated by using MPI non-blocking communication operations). This strategy avoids deadlock.

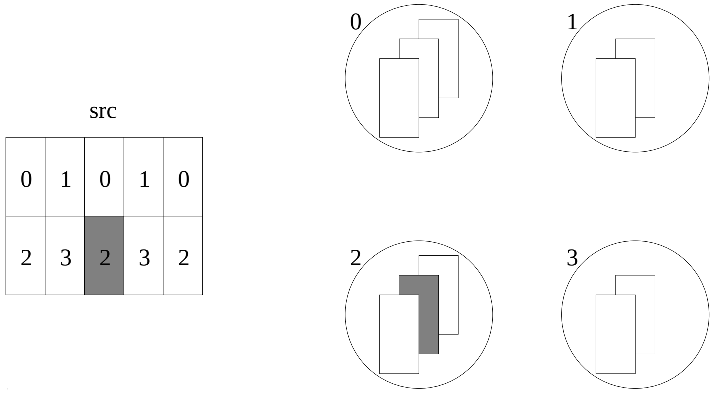

Figure 1: Distribution of array *src* onto a grid of four processors

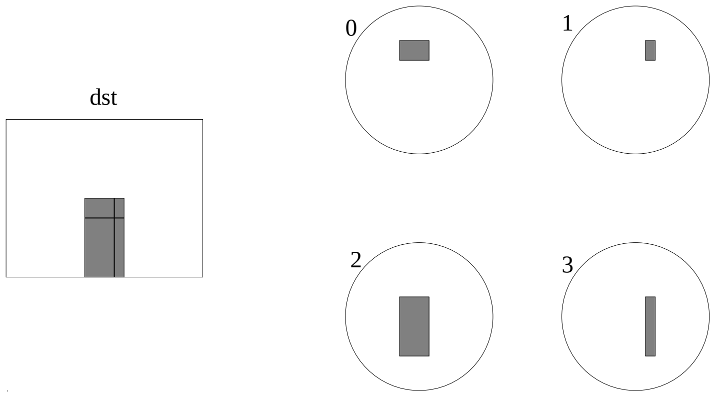

Figure 2: Distribution of section of *dst* array corresponding to highlighted block of *src* array

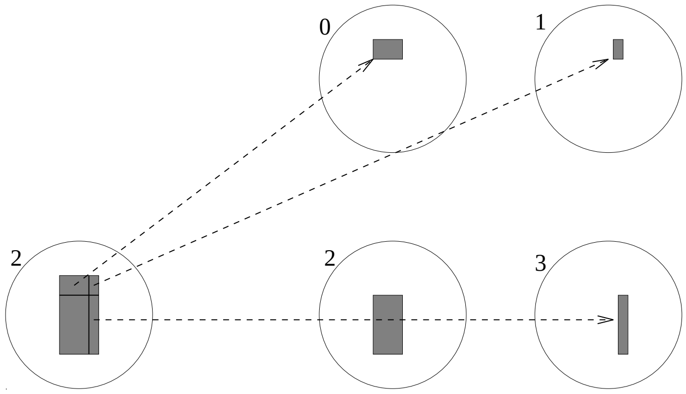

Figure 3: Block send operations required to redistribute highlighted block of *src* array