Advanced Parallel Programming
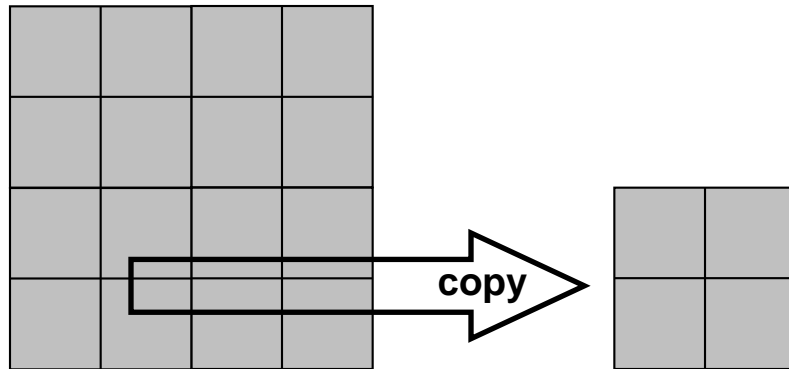
Overview of MPI-IO Exercises

Dr David Henty
HPC Training and Support
d.henty@epcc.ed.ac.uk
+44 131 650 5960

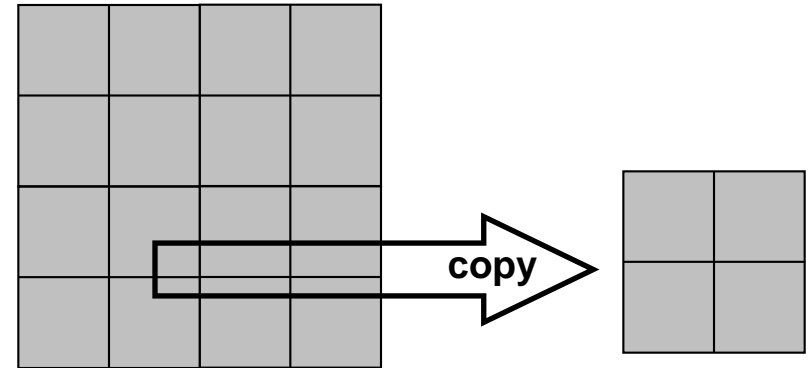# Stand-Alone Exercise

- We will
  - compile some existing source
  - run it on one and many processors
    - on the front-end and the back-end
  - implement master IO for reading into a block decomposition
    - using global broadcast then copy-back on individual processes
    - by copying appropriate data to a buffer on the master and sending

- Later on
  - extend this to using appropriate datatypes to avoid data copying
  - use above datatypes to achieve same result with MPI-IO
  - extend to do general block-cyclic decompositions with MPI-IO

- Input and output data files can be viewed as pictures
  - to make debugging easier!

- See PDF exercise sheet for full details

MSc Students: Extending MPP Coursework

- Probably more interesting to put MPI-IO into your own code
  - e.g. your 2D-decomposed MPP coursework exercise

- Issues
  - file formats are different: must replace `pgmread` and `pgmwrite` with `ioread` and `iowrite`
  - need special programs to view the files: `cioview` or `fioview`
  - must have specific names to be visualised: use `creatfilename`

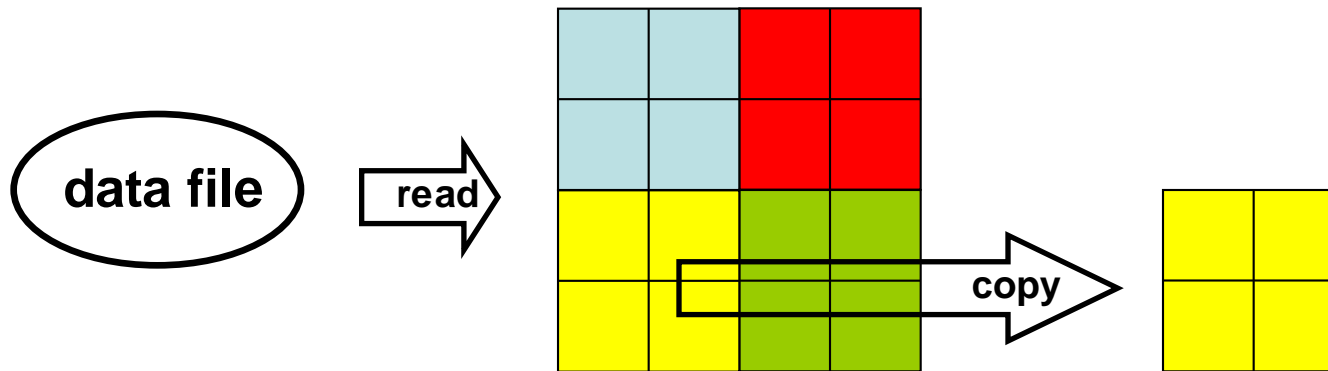- See additional exercise sheet `mpiio-mpp.pdf` for details

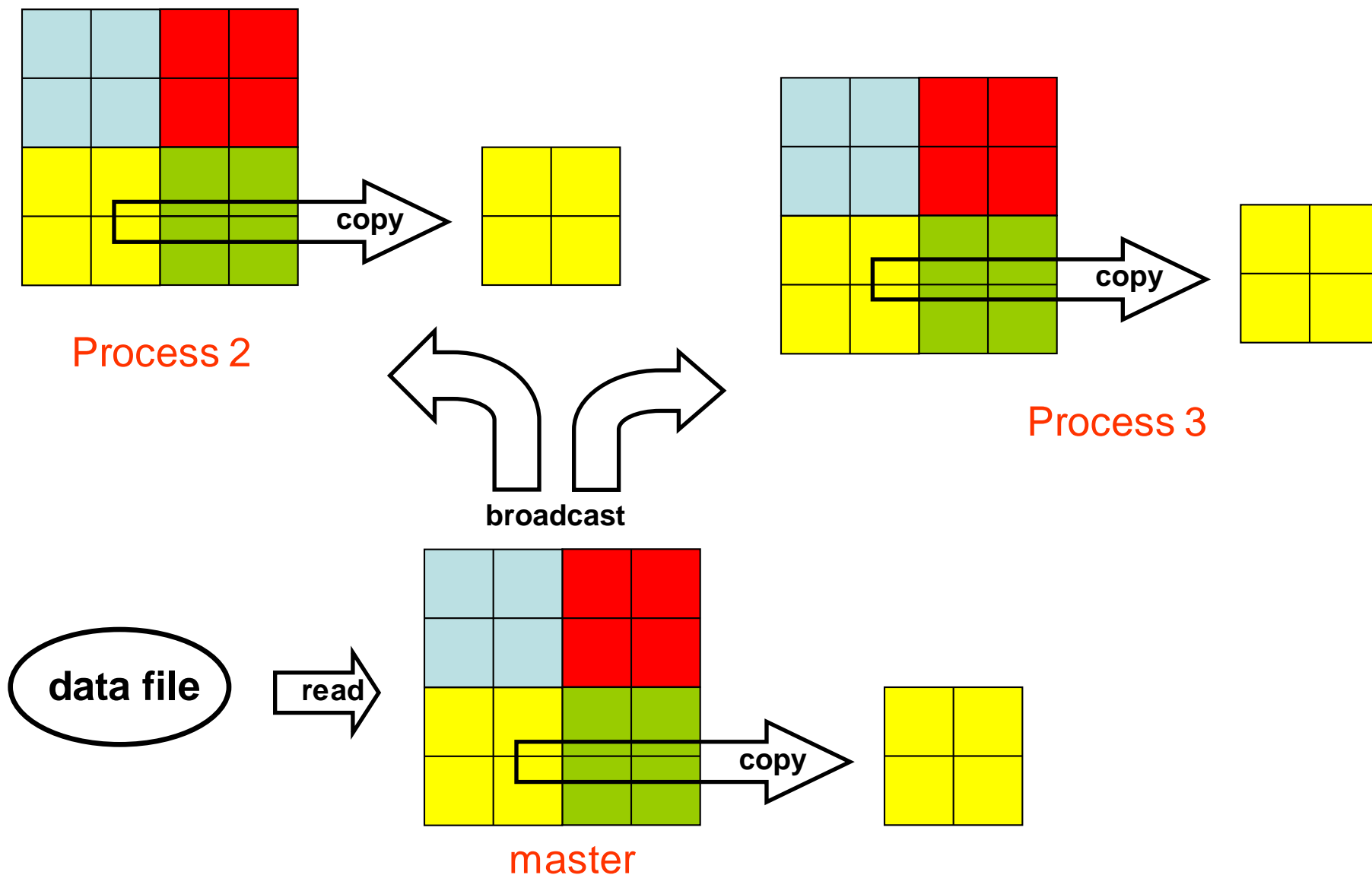# IO strategy 1: no data on workers
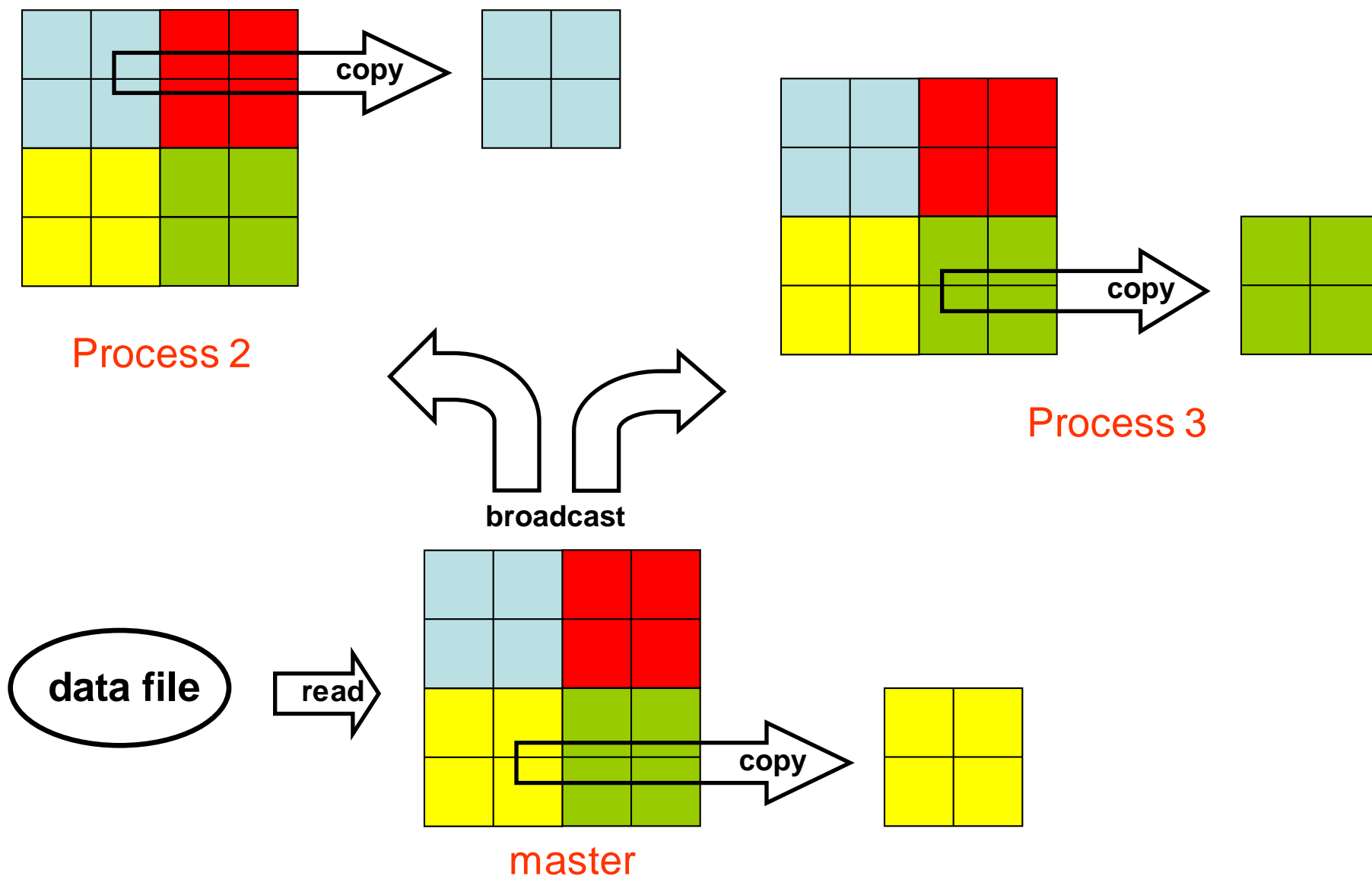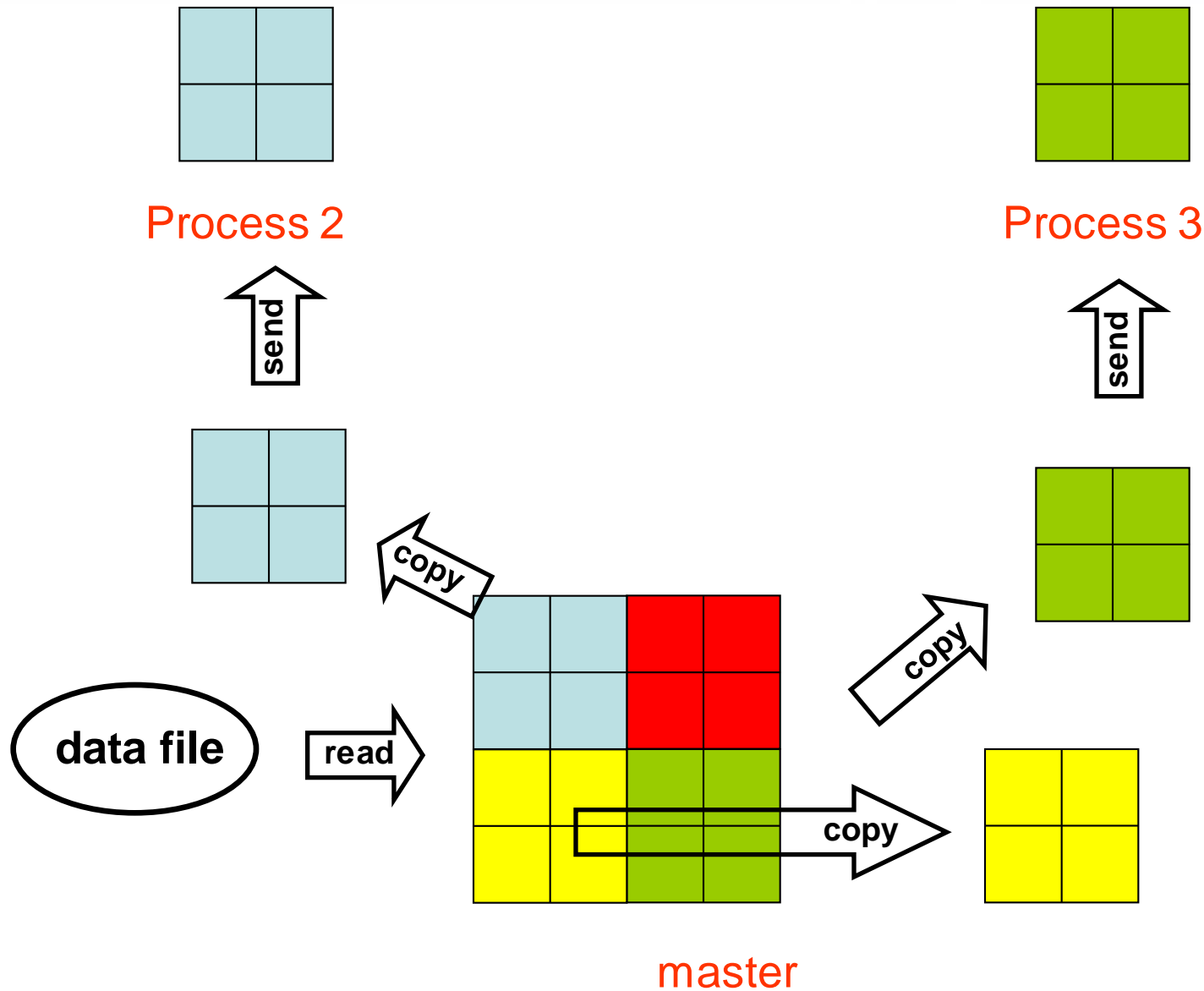

Process 2

Process 3

data file → read

copy

master

Process 2

copy

broadcast

Process 3

copy

data file

read

copy

master

Process 2

Process 3

copy

copy

broadcast

data file

read

copy

master

IO strategy 4: standard master IO

Process 2

Process 3

send

send

copy

copy

copy

data file → read

master

IO strategy 5: vector datatype

Process 2

Process 3

subarray2

subarray3

data file

read

copy

master

Using MPI-IO
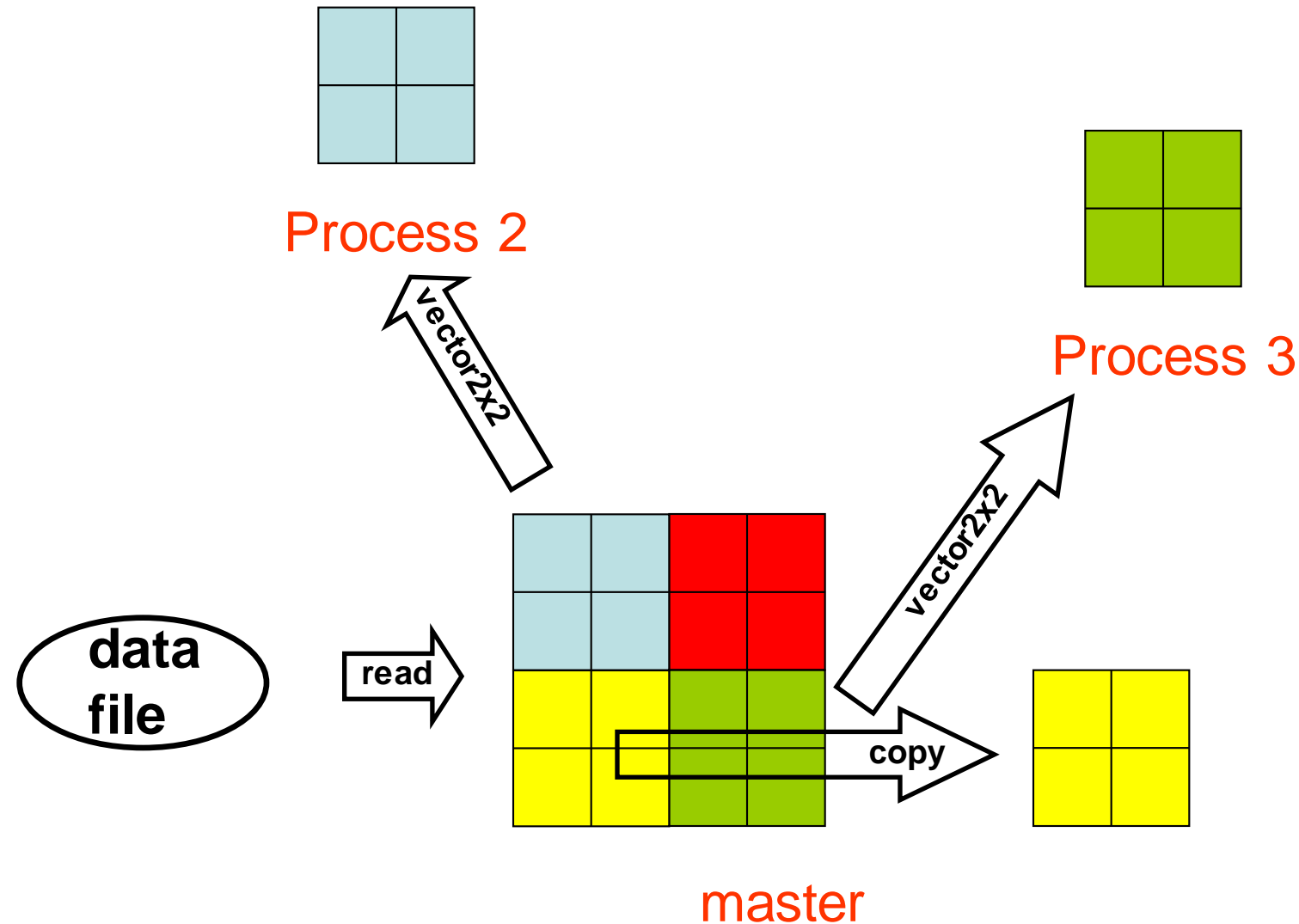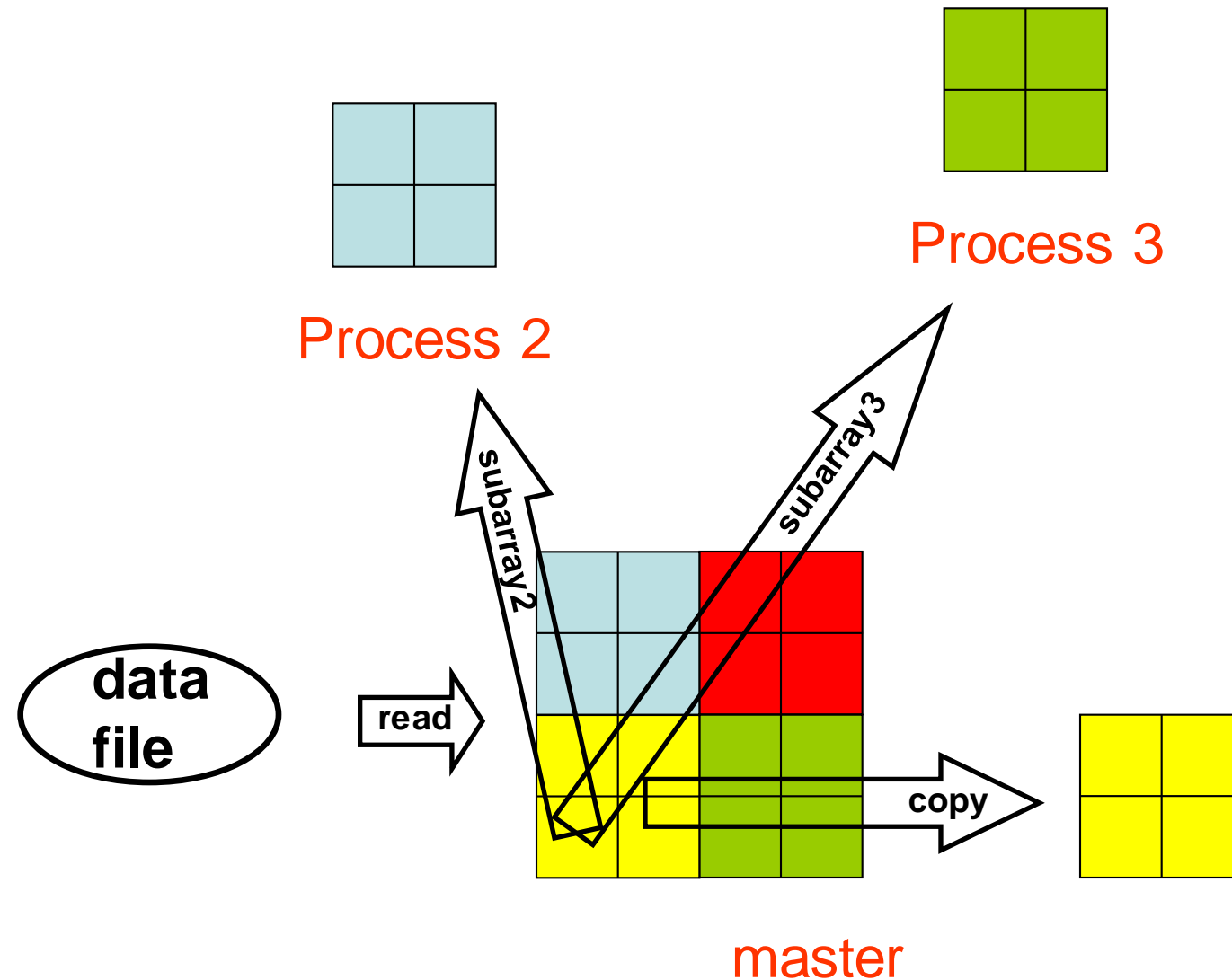
- Pass your derived datatypes to appropriate MPI-IO calls
  - use subarrays with `disp = 0`
  - or a vector with non-zero values of `disp`

Template Code (Fortran)

```fortran
! define subarray datatype for this process

INTEGER(KIND=MPI_OFFSET_KIND) disp

disp = 0

call MPI_File_open(…, fh, ierr)

call MPI_File_set_view(fh, disp, MPI_REAL,
        subarray, 'native', MPI_INFO_NULL, ierr)


call MPI_File_read_all(fh,buf,count,MPI_REAL,…)

call MPI_File_close(fh, ierr)
```

```
/* define subarray datatype for this process */

 MPI_File fh;

 MPI_File_open(…, &fh);

 MPI_File_set_view(fh, 0, MPI_FLOAT, subarray,

                     "native", MPI_INFO_NULL);


  MPI_File_read_all(fh, buf, count, MPI_FLOAT, …)

  MPI_File_close(&fh);
```