Parallel I/O

1. General problems with I/O in parallel computing

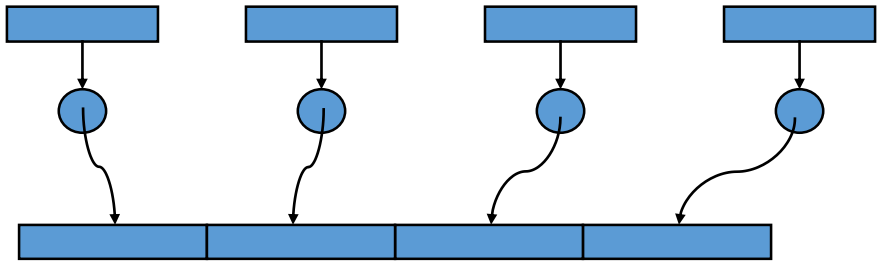
- How I/O works serially.
- Why seek could be a problem in parallel computing.
- How to optimize I/O operations in general.

2. File views

Use **Mpi_File_open** to open a shared file for all processes:

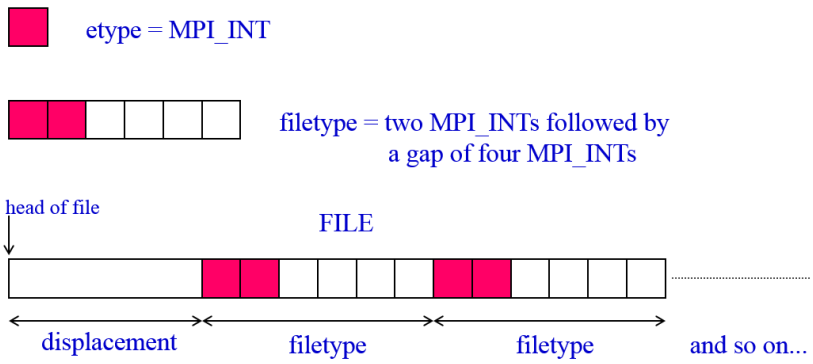
MPI_File_open(MPI_COMM_WORLD, "testfile.bin", MPI_MODE_CREATE | MPI_MODE_WRONLY, MPI_INFO_NULL, &thefile);

MPI_File_set_view assigns regions of the file to separate processes.



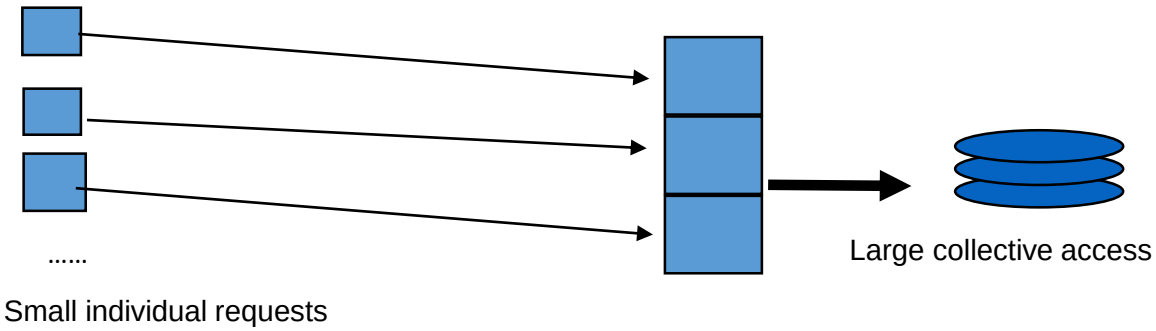
MPI_Set_view(MPI_File fh, MPI_Offset disp, MPI_Datatype etype, MPI_Datatype filetype, char *datarep, MPI_Info info)

- View can be changed any number of times during a program
- All file access done in unites of **etype**; **filetype** must be equal to or be derived from **etype**



3. Basic Collective I/O

- Critical optimization in Parallel I/O.
- 2 phase I/O: communication first, then I/O. The idea is to build large blocks first then read/writes in I/O system will be large.



When using **MPI_File_read_all and MPI_File_write_all :**

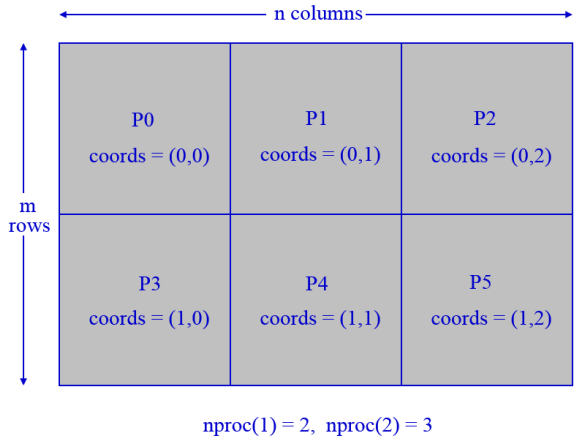
- All processes in the group will call this function.
- Backend can merge the requests from different processes and service the merged request efficiently. (Particularly effective when the access of different processes are noncontiguous and interleaved).

Note: Look at example (1.cpp) in the solution attached.

If 3 processes are used in this example, the file will be split like below:

- When you read data on a process P, it will always advance to the correct block of that process
- **Question**: Why is this more efficient than writing all data blocks of a process on a contiguous space.

4. N-dimensional array stored in a file




```
int gsizes[2], distribs[2], dargs[2], psizes[2];
gsizes[0] = m;    /* no. of rows in global array */
gsizes[1] = n;    /* no. of columns in global array*/
distribs[0] = MPI_DISTRIBUTE_BLOCK;
distribs[1] = MPI_DISTRIBUTE_BLOCK;
dargs[0] = MPI_DISTRIBUTE_DFLT_DARG;
dargs[1] = MPI_DISTRIBUTE_DFLT_DARG;
psizes[0] = 2; /* no. of processes in vertical dimension of process grid */
psizes[1] = 3; /* no. of processes in horizontal dimension of process grid */
```

MPI_Type_create_darray(6, rank, 2, gsizes, distribs, dargs, psizes, MPI_ORDER_C, MPI_FLOAT, &**filetype**);

// (num processes, rank, num dimensions, num items per dimensions, distribs =way in which array is distributed in each dimension can be block, cyclic or block-cyclic, default, array of procs in each dimension.

Exercises:

- Modify 1.cpp and use: MPI_File_set_view on the new file to use the new filetype created and perform a few read/write operations.
- Modify the code of lab 4 (“GameOfLife” app) and make each process write statistics to a shared file per frame regarding population.

5. Testing & Analysis

Check **Test & Analysis.ppt to** see compared performance between different I/O methods.