I/O Consistency Semantics

- The consistency semantics specify the results when multiple processes access a common file and one or more processes write to the file
- MPI guarantees stronger consistency semantics if the communicator used to open the file accurately specifies all the processes that are accessing the file, and weaker semantics if not
- The user can take steps to ensure consistency when MPI does not automatically do so

Example 1

- File opened with **`MPI_COMM_WORLD`**. Each process writes to a *separate* region of the file and reads back only what it wrote.

Process 0	Process 1
`MPI_File_open(MPI_COMM_WORLD,…)`	`MPI_File_open(MPI_COMM_WORLD,…)`
`MPI_File_write_at(off=0,cnt=100)`	`MPI_File_write_at(off=100,cnt=100)`
`MPI_File_read_at(off=0,cnt=100)`	`MPI_File_read_at(off=100,cnt=100)`

  - MPI guarantees that the data will be read correctly

Example 2

- Same as example 1, except that each process wants to read what the *other* process wrote (overlapping accesses)
- In this case, MPI does *not* guarantee that the data will automatically be read correctly

Process 0

```
/* incorrect program */
MPI_File_open(MPI_COMM_WORLD,…)
MPI_File_write_at(off=0,cnt=100)
MPI_Barrier
MPI_File_read_at(off=100,cnt=100)
```

Process 1

```
/* incorrect program */
MPI_File_open(MPI_COMM_WORLD,…)
MPI_File_write_at(off=100,cnt=100)
MPI_Barrier
MPI_File_read_at(off=0,cnt=100)
```

- **In the above program, the read on each process is not guaranteed to get the data written by the other process!**

Example 2 contd.

- The user must take extra steps to ensure correctness
- There are three choices:
  - set atomicity to true
  - close the file and reopen it
  - ensure that no write sequence on any process is concurrent with any sequence (read or write) on another process

Example 2, Option 1
Set atomicity to true

Process 0	Process 1
`MPI_File_open(MPI_COMM_WORLD,…)`	`MPI_File_open(MPI_COMM_WORLD,…)`
`MPI_File_set_atomicity(fh1,1)`	`MPI_File_set_atomicity(fh2,1)`
`MPI_File_write_at(off=0,cnt=100)`	`MPI_File_write_at(off=100,cnt=100)`
`MPI_Barrier`	`MPI_Barrier`
`MPI_File_read_at(off=100,cnt=100)`	`MPI_File_read_at(off=0,cnt=100)`

Example 2, Option 2
Close and reopen file

Process 0	Process 1
`MPI_File_open(MPI_COMM_WORLD,…)`	`MPI_File_open(MPI_COMM_WORLD,…)`
`MPI_File_write_at(off=0,cnt=100)`	`MPI_File_write_at(off=100,cnt=100)`
`MPI_File_close`	`MPI_File_close`
`MPI_Barrier`	`MPI_Barrier`
`MPI_File_open(MPI_COMM_WORLD,…)`	`MPI_File_open(MPI_COMM_WORLD,…)`
`MPI_File_read_at(off=100,cnt=100)`	`MPI_File_read_at(off=0,cnt=100)`

Example 2, Option 3

- *Ensure that no write sequence on any process is concurrent with any sequence (read or write) on another process*
- a sequence is a set of operations between any pair of open, close, or file_sync functions
- a write sequence is a sequence in which any of the functions is a write operation

Example 2, Option 3

Process 0	Process 1
`MPI_File_open(MPI_COMM_WORLD,…)`	`MPI_File_open(MPI_COMM_WORLD,…)`
`MPI_File_write_at(off=0,cnt=100)`	
`MPI_File_sync`	`MPI_File_sync /*collective*/`
`MPI_Barrier`	`MPI_Barrier`
`MPI_File_sync /*collective*/`	`MPI_File_sync`
	`MPI_File_write_at(off=100,cnt=100)`
`MPI_File_sync /*collective*/`	`MPI_File_sync`
`MPI_Barrier`	`MPI_Barrier`
`MPI_File_sync`	`MPI_File_sync /*collective*/`
`MPI_File_read_at(off=100,cnt=100)`	`MPI_File_read_at(off=0,cnt=100)`
`MPI_File_close`	`MPI_File_close`

Example 3

- Same as Example 2, except that each process uses **`MPI_COMM_SELF`** when opening the common file
- The only way to achieve consistency in this case is to ensure that no write sequence on any process is concurrent with any write sequence on any other process.

Example 3

Process 0	Process 1
`MPI_File_open(MPI_COMM_SELF,…)`	`MPI_File_open(MPI_COMM_SELF,…)`
`MPI_File_write_at(off=0,cnt=100)`	
`MPI_File_sync`	
`MPI_Barrier`	`MPI_Barrier`
	`MPI_File_sync`
	`MPI_File_write_at(off=100,cnt=100)`
	`MPI_File_sync`
`MPI_Barrier`	`MPI_Barrier`
`MPI_File_sync`	
`MPI_File_read_at(off=100,cnt=100)`	`MPI_File_read_at(off=0,cnt=100)`
`MPI_File_close`	`MPI_File_close`