MFIX Distributed IO Documentation

The ability to use distributed IO in MFIX has been added (each processor writes its own section of the grid to its own file). One advantage of this approach is that the arrays do not need to be copied to the head node for input/output processing. The feature has not been fully developed, since some researchers in the field suggested that writing one data file per processor would lead to poor performance on the disk system. A project is underway to investigate this issue.

Current State of Distributed IO in MFIX

By setting a flag in MFIX.DAT, you can turn on the distributed IO option. Starting with a 'NEW' run type, you can then do 'RESTART_1' and 'RESTART_2' runs as usual. There is a restriction - you can not change the number of processors or the NODESI, NODESJ, and NODESK values. It is also possible to restart a run with distributed processing, even if the run was started without distributed IO. See below for details.

None of the post-processing routines available for MFIX have been modified to read the files created using distributed IO. POST_MFIX is able to convert the distributed IO files into normal MFIX output. The distributed IO files option is not useful at this time because of the lack of post processing tools that can read the files.

Usage

Distributed IO is activated in MFIX by setting the following variable in mfix.dat

- bdist_io = T

Running the code on N processors will create files of the form (assuming a run name = BUB):

- BUB_00000.RES
- BUB_00001.RES
- BUB_0000N.RES
- If N = 100, the last file would be named : BUB_00100.RES
- The SP1, SP2, SP3, ... , SPB files all have the same naming conventions.

Also created are files which have processor node information:

- p_info_00000.txt
- p_info_00001.txt
- p_info_0000N.txt
- These files are needed by post_mfix to combine the distributed IO files into standard MFIX output files.

Creating standard MFIX output files from distributed IO files

POST_MFIX requires a restart file with the standard naming convention in order to startup. Before running post_mfix, create that file by copying the first RES file.

In the example above:

- cp BUB_00000.RES BUB.RES

Run post_mfix. You must choose the option: "10 - run scavenger code". Sample execution shown below:

```
Enter menu selection > 10

enter number of processors

8

what files should be stitched together ?


  -1 : RES and all SPX

   0 : RES only

 k>0 : The kth SPx file (k=1 to 11)

 -1
```

The number of processors must be the same as when MFIX was run. Additionally, the "p_info*.txt" files must exist. The code will combine the distributed output files and create files named:

- BUB_SCAV.RES
- BUB_SCAV.SP1
- BUB_SCAV.SP2
- Etc.

These files can be post-processed as usual.

Activating distributed IO on a run started with standard MFIX IO

This can be done only for 'RESTART_2' runs.

Steps:

- Set the following variables to .true. in mfix.dat:
  - bdist_io = T
  - bStart_with_one_RES = T
- run mfix.exe
- when finished, create the distributed "_00000.RES" file from the standard RES file:
  - cp   BUB.RES   BUB_00000.RES
- in subsequent runs, set
  - bStart_with_one_RES = F

Changing the number of processors on a distributed IO run:

The only way this can currently be done is to:

- Create standard MFIX files from the distributed IO files (see above)
- Optionally, rename the "_SCAV" files created as desired
- Activate distributed IO on these files (see above).
  - Note: you must do a RESTART_2 run.