Carrying out restart runs with `mrun`

PALM group

Institute of Meteorology and Climatology, Leibniz Universität Hannover

last update: 21st September 2015

Carrying out restart runs with mrun

Definition of “restart run“

- A **“restart run“** is a model run, which starts with an initial condition given by the simulated flow at the end of a previous (restart or initial) run.

Carrying out restart runs with mrun

Definition of “restart run“

- A **“restart run“** is a model run, which starts with an initial condition given by the simulated flow at the end of a previous (restart or initial) run.
- In order to carry out a restart run, a file has to be written at the end of the previous run, which contains the values of all flow variables at the last time step. This file has to be read at the beginning of the restart run.

Definition of “restart run“

- A **“restart run“** is a model run, which starts with an initial condition given by the simulated flow at the end of a previous (restart or initial) run.
- In order to carry out a restart run, a file has to be written at the end of the previous run, which contains the values of all flow variables at the last time step. This file has to be read at the beginning of the restart run.
- Initial and respective restart runs form a so called **job chain**.

Reasons for Restart Runs

- The maximum job time is generally limited by the queuing system:
  - simulations must be split into several parts

Reasons for Restart Runs

- The maximum job time is generally limited by the queuing system:
  - simulations must be split into several parts
- The user wants to carry out several runs on the basis of the same initial temporal development:
  - the initial phase needs to be simulated only once, all runs start from the end point of this initial phase by reading the flow field data written at the end of the initial run

Carrying Out Restart Runs With `mrun`

Concerning `mrun`, the first thing required to enable restart runs is to use the additional activating string "`restart`" in the `mrun`-call for the initial run:

```
mrun -d test ...  -r "d3# restart"
```

This will have the following effects:

Carrying Out Restart Runs With mrun

Concerning mrun, the first thing required to enable restart runs is to use the additional activating string "restart" in the mrun-call for the initial run:

```
mrun -d test ...  -r "d3# restart"
```

This will have the following effects:

- At the end of the run, all necessary variables will bei written as binary data to the local file BINOUT. This is caused by an entry in the configuration file

  ```
  %write_binary true restart
  ```

  which sets the environment variable write_binary, which is in turn read by PALM from the local file ENVPAR created by mrun.

Carrying Out Restart Runs With mrun

Concerning mrun, the first thing required to enable restart runs is to use the additional activating string "restart" in the mrun-call for the initial run:

```
mrun -d test ...  -r "d3# restart"
```

This will have the following effects:

- At the end of the run, all necessary variables will bei written as binary data to the local file BINOUT. This is caused by an entry in the configuration file

  ```
  %write_binary true restart
  ```

  which sets the environment variable write_binary, which is in turn read by PALM from the local file ENVPAR created by mrun.

- This binary file will be permanently stored in case that an appropriate file connection statement exists

  ```
  BINOUT out:loc:flpe restart ~/palm/current_version/JOBS/$fname/RESTART _d3d
  ```

Carrying Out Restart Runs With mrun

Concerning mrun, the first thing required to enable restart runs is to use the additional activating string "restart" in the mrun-call for the initial run:

```
mrun -d test ...  -r "d3# restart"
```

This will have the following effects:

- At the end of the run, all necessary variables will bei written as binary data to the local file BINOUT. This is caused by an entry in the configuration file

  ```
  %write_binary true restart
  ```

  which sets the environment variable write_binary, which is in turn read by PALM from the local file ENVPAR created by mrun.

- This binary file will be permanently stored in case that an appropriate file connection statement exists

  ```
  BINOUT out:loc:flpe restart ~/palm/current_version/JOBS/$fname/RESTART _d3d
  ```

- If, during the run, PALM detects that the simulation cannot be finished due to limited job time, it tells mrun (by creating a local file named CONTINUE_RUN) that a restart job has to be started. mrun will then automatically start such a job by submitting the command

  ```
  mrun -d test ...  -r "d3f restart"
  ```

  on the **local host**. Options of this command are nearly the same as of the initial run, but every sharp character ("#") in the activating strings is replaced by an "f".

Carrying Out Restart Runs With mrun

Concerning mrun, the first thing required to enable restart runs is to use the additional activating string "restart" in the mrun-call for the initial run:

```
mrun -d test ...  -r "d3# restart"
```

This will have the following effects:

- At the end of the run, all necessary variables will bei written as binary data to the local file BINOUT. This is caused by an entry in the configuration file

  ```
  %write_binary true restart
  ```

  which sets the environment variable write_binary, which is in turn read by PALM from the local file ENVPAR created by mrun.

- This binary file will be permanently stored in case that an appropriate file connection statement exists

  ```
  BINOUT out:loc:flpe restart ~/palm/current_version/JOBS/$fname/RESTART _d3d
  ```

- If, during the run, PALM detects that the simulation cannot be finished due to limited job time, it tells mrun (by creating a local file named CONTINUE_RUN) that a restart job has to be started. mrun will then automatically start such a job by submitting the command

  ```
  mrun -d test ...  -r "d3f restart"
  ```

  on the **local host**. Options of this command are nearly the same as of the initial run, but every sharp character ("#") in the activating strings is replaced by an "f".

This effects the activation of file connections for the restart job!

Input Files Necessary For Restart Jobs

File connection statements for input files from the default `.mrun.config` file:

```
PARIN   in:job          d3#   $base_data/$fname/INPUT      _p3d
PARIN   in:job          d3f   $base_data/$fname/INPUT      _p3df
BININ   in:loc:flpe     d3f   $base_data/$fname/RESTART    _d3d
```

Input Files Necessary For Restart Jobs

File connection statements for input files from the default `.mrun.config` file:

```
PARIN   in:job          d3#   $base_data/$fname/INPUT      _p3d
PARIN   in:job          d3f   $base_data/$fname/INPUT      _p3df
BININ   in:loc:flpe     d3f   $base_data/$fname/RESTART    _d3d
```

- For the restart job, the model receives a different parameter file than for the initial job (e.g. `example_cbl_p3df` instead of `example_cbl_p3d`).

  The parameter file for the restart job is nearly the same as for the initial run, but it must contain the parameter setting

  ```
  initializing_actions = 'read_restart_data'
  ```

  in the `&inipar`-NAMELIST-group. All other `&inipar`-parameter-settings are ignored!

  `&d3par`-parameter values can freely be changed compared with the parameter file for the initial run.

Input Files Necessary For Restart Jobs

File connection statements for input files from the default `.mrun.config` file:

```
PARIN   in:job          d3#   $base_data/$fname/INPUT      _p3d
PARIN   in:job          d3f   $base_data/$fname/INPUT      _p3df
BININ   in:loc:flpe     d3f   $base_data/$fname/RESTART    _d3d
```

- For the restart job, the model receives a different parameter file than for the initial job (e.g. `example_cbl_p3df` instead of `example_cbl_p3d`).

  The parameter file for the restart job is nearly the same as for the initial run, but it must contain the parameter setting

  ```
  initializing_actions = 'read_restart_data'
  ```

  in the `&inipar`-NAMELIST-group. All other `&inipar`-parameter-settings are ignored!

  `&d3par`-parameter values can freely be changed compared with the parameter file for the initial run.

- Input binary data file (`BININ`) is necessary (and available) only for restart jobs

Output File Handling in Restart Jobs

Example for output file connection statements from the default `.mrun.config` file:

```
RUN_CONTROL    out:loc:tr     d3#    $base_data/$fname/MONITORING    _rc
RUN_CONTROL    out:loc:tra    d3f    $base_data/$fname/MONITORING    _rc
```

In case of restart jobs, the contents of many local output files are appended to the respective permanent files from the initial or previous run by using the `tra` file attribute.

Output File Handling in Restart Jobs

Example for output file connection statements from the default `.mrun.config` file:

```
RUN_CONTROL    out:loc:tr    d3#    $base_data/$fname/MONITORING    _rc
RUN_CONTROL    out:loc:tra   d3f    $base_data/$fname/MONITORING    _rc
```

In case of restart jobs, the contents of many local output files are appended to the respective permanent files from the initial or previous run by using the `tra` file attribute.

File connection statement example for appending netCDF files when PALM is running on a remote host:

```
DATA_1D_PR_NETCDF  in:loc     prf      $base_data/$fname/OUTPUT _pr nc
DATA_1D_PR_NETCDF  out:loc    pr#:prf  $base_data/$fname/OUTPUT _pr nc
DATA_1D_PR_NETCDF  out:loc:tr pr#:prf  $base_data/$fname/OUTPUT _pr nc
```

The netCDF file from the respective previous run has to be provided as an INPUT file.

Therefore, if running PALM on a remote host, a copy of this data file must be additionally stored on the remote host (second statement). On the local host, each run creates a new file (cycle) which contains the complete data from the current run and all previous runs.

Handling of Large Binary Data Files

- Typically, the binary restart files are very large, so that they cannot be stored in the user's home-directory because of limited disk quotas. Also, hard disks where `/home` is stored are typically very slow, so that the copy process needs very long time.

Handling of Large Binary Data Files

- Typically, the binary restart files are very large, so that they cannot be stored in the user's home-directory because of limited disk quotas. Also, hard disks where `/home` is stored are typically very slow, so that the copy process needs very long time.
- Using the file attribute `fl` (abbreviation for german "Fortsetzungslauf") in the output file connection statement causes `mrun` to copy the local file to a special directory, which can be defined in the configuration file by the environment variable `tmp_data_catalog`. The permanent file described in the connection statement is also created, but it is **empty**.

Handling of Large Binary Data Files

- Typically, the binary restart files are very large, so that they cannot be stored in the user's home-directory because of limited disk quotas. Also, hard disks where `/home` is stored are typically very slow, so that the copy process needs very long time.
- Using the file attribute `fl` (abbreviation for german "Fortsetzungslauf") in the output file connection statement causes `mrun` to copy the local file to a special directory, which can be defined in the configuration file by the environment variable `tmp_data_catalog`. The permanent file described in the connection statement is also created, but it is **empty**.
- At the end of the job, the second last cycle of the respective file with attribute `fl` is automatically deleted by `mrun` from the `tmp_data_catalog` in order to save disk space. This can be switched off with `mrun`-option "`-k`"(keep data from previous run).

Handling of Large Binary Data Files

- Typically, the binary restart files are very large, so that they cannot be stored in the user's home-directory because of limited disk quotas. Also, hard disks where `/home` is stored are typically very slow, so that the copy process needs very long time.
- Using the file attribute `fl` (abbreviation for german "Fortsetzungslauf") in the output file connection statement causes `mrun` to copy the local file to a special directory, which can be defined in the configuration file by the environment variable `tmp_data_catalog`. The permanent file described in the connection statement is also created, but it is **empty**.
- At the end of the job, the second last cycle of the respective file with attribute `fl` is automatically deleted by `mrun` from the `tmp_data_catalog` in order to save disk space. This can be switched off with `mrun`-option "`-k`"(keep data from previous run).

Example:

```
%base_data          ~/palm/current_version/JOBS
%tmp_data_catalog   /gfs1/work/niksiraa/palm_restart_data

BINOUT   out:loc:flpe   restart   $base_data/$fname/RESTART   _d3d
```

Handling of Large Binary Data Files

- Typically, the binary restart files are very large, so that they cannot be stored in the user's home-directory because of limited disk quotas. Also, hard disks where `/home` is stored are typically very slow, so that the copy process needs very long time.
- Using the file attribute `fl` (abbreviation for german "Fortsetzungslauf") in the output file connection statement causes `mrun` to copy the local file to a special directory, which can be defined in the configuration file by the environment variable `tmp_data_catalog`. The permanent file described in the connection statement is also created, but it is **empty**.
- At the end of the job, the second last cycle of the respective file with attribute `fl` is automatically deleted by `mrun` from the `tmp_data_catalog` in order to save disk space. This can be switched off with `mrun`-option "`-k`"(keep data from previous run).

Example:

```
%base_data          ~/palm/current_version/JOBS
%tmp_data_catalog   /gfs1/work/niksiraa/palm_restart_data

BINOUT   out:loc:flpe   restart   $base_data/$fname/RESTART   _d3d
```

Files (directories) created when using `-d example_cbl`:

```
/gfs1/work/niksiraa/palm_restart_data/example_cbl_d3d
~/palm/current_version/JOBS/example/RESTART/example_cbl_d3d # empty file (directory)
```

Handling of Large Binary Data Files

- Typically, the binary restart files are very large, so that they cannot be stored in the user's home-directory because of limited disk quotas. Also, hard disks where `/home` is stored are typically very slow, so that the copy process needs very long time.
- Using the file attribute `fl` (abbreviation for german "Fortsetzungslauf") in the output file connection statement causes `mrun` to copy the local file to a special directory, which can be defined in the configuration file by the environment variable `tmp_data_catalog`. The permanent file described in the connection statement is also created, but it is **empty**.
- At the end of the job, the second last cycle of the respective file with attribute `fl` is automatically deleted by `mrun` from the `tmp_data_catalog` in order to save disk space. This can be switched off with `mrun`-option "`-k`"(keep data from previous run).

Example:

```
%base_data          ~/palm/current_version/JOBS
%tmp_data_catalog   /gfs1/work/niksiraa/palm_restart_data

BINOUT   out:loc:flpe   restart   $base_data/$fname/RESTART   _d3d
```

Files (directories) created when using `-d example_cbl`:

```
/gfs1/work/niksiraa/palm_restart_data/example_cbl_d3d
~/palm/current_version/JOBS/example/RESTART/example_cbl_d3d # empty file (directory)
```

Concerning input files, `mrun` always determines the current cycle number to be used from the contents of the directory given by the file connection statement!

Checking the Restart Job Execution

- essentially by looking at the messages in the job protocol file:

```
  *** execution starts in directory
      "/gfs1/work/nikleboe/nikleboe.21239"
  ------------------------------------------------------------------------
.
.
    run will be terminated due to user settings of
    restart_time / dt_restart
    new restart time is:  3600.  s
.
.
  ------------------------------------------------------------------------
  *** execution finished
.
.
 ------------------------------------------------------------------------
  *** all OUTPUT-files saved

  *** initiating restart-run on "130.75.105.111" using command:
      mrun -c.mrun.config -dexample_cbl -hlccrayh -Hlcmuk -m1500 -t3600 -qmpp1testq -R130.75.105.111 -Uboeske ...
  ------------------------------------------------------------------------
*** ssh will be used to initiate restart-runs!
.
.
*** MRUN  2.1 Rev: 1358 $
    will be executed.     Please wait ...
#--------------------------------------------------------------------#
| MRUN  2.1 Rev: 1358 $                  Thu Jun 31 14:09:30 CEST 2014 |
|                                                                     |
| called on:               vaudaire                                   |
.
.
| Files to be compiled:                                               |
| palm.f90 user_example.f90                                           |
#--------------------------------------------------------------------#
.
.
  ------------------------------------------------------------------------
  *** restart-run initiated

 --> all actions finished

     Bye, bye nikleboe !!
```

In this example, restart time has been set manually by the user.

Setting the Restart Time Manually

- By default, PALM checks after every timestep, if enough time remains from the job's cpu limit to carry out the next timestep:

  ("total job time" - "time already consumed") <= termination_time_needed
  (as given by mrun-option -t ...) (as given by parameter in &d3par-NAMELIST)

Setting the Restart Time Manually

- By default, PALM checks after every timestep, if enough time remains from the job's cpu limit to carry out the next timestep:

  ("total job time" - "time already consumed") <= termination_time_needed
  (as given by mrun-option -t ...) (as given by parameter in &d3par-NAMELIST)

- termination_time_needed has to include the cpu time needed before running PALM (e.g. for compilation, copying of input data, etc.; default value: 300 s)!

Setting the Restart Time Manually

- By default, PALM checks after every timestep, if enough time remains from the job's cpu limit to carry out the next timestep:

  ("total job time" - "time already consumed") <= termination_time_needed
  (as given by mrun-option -t ...) (as given by parameter in &d3par-NAMELIST)

- termination_time_needed has to include the cpu time needed before running PALM (e.g. for compilation, copying of input data, etc.; default value: 300 s)!

  Warning:

  "total job time" <= termination_time_needed,
  forces a restart after every timestep!

Setting the Restart Time Manually

- By default, PALM checks after every timestep, if enough time remains from the job's cpu limit to carry out the next timestep:

  ("total job time" - "time already consumed") <= termination_time_needed
  (as given by mrun-option -t ...) (as given by parameter in &d3par-NAMELIST)

- termination_time_needed has to include the cpu time needed before running PALM (e.g. for compilation, copying of input data, etc.; default value: 300 s)!

  Warning:

  "total job time" <= termination_time_needed,
  forces a restart after every timestep!

- &d3par-parameters restart_time and dt_restart can be used to set restart time(s) manually.

Setting the Restart Time Manually

- By default, PALM checks after every timestep, if enough time remains from the job's cpu limit to carry out the next timestep:

  ("total job time" - "time already consumed") <= termination_time_needed
  (as given by mrun-option -t ...) (as given by parameter in &d3par-NAMELIST)

- termination_time_needed has to include the cpu time needed before running PALM (e.g. for compilation, copying of input data, etc.; default value: 300 s)!

  Warning:

  "total job time" <= termination_time_needed,
  forces a restart after every timestep!

- &d3par-parameters restart_time and dt_restart can be used to set restart time(s) manually.

- In case of manually setting the restart time, the default checking (see above) is still active and a restart will be automatically forced if the job reaches its cpu limit, even if the manually set restart time has not been reached!

Starting Restart Jobs Manually

- After a job has finished (`end_time` has been reached), the user can submit a restart job manually (provided that restart data have been saved) by entering:

```
mrun ...  -r "d3f ..." ...
```

or

```
mrun ...  -r "d3f restart ..." ...
```

Starting Restart Jobs Manually

- After a job has finished (`end_time` has been reached), the user can submit a restart job manually (provided that restart data have been saved) by entering:

```
mrun ...  -r "d3f ..." ...
```

or

```
mrun ...  -r "d3f restart ..." ...
```

- Remember to increase the value of `end_time` in the parameter file before submitting the job.

Starting Restart Jobs Manually

- After a job has finished (`end_time` has been reached), the user can submit a restart job manually (provided that restart data have been saved) by entering:

```
mrun ...  -r "d3f ..." ...
```

or

```
mrun ...  -r "d3f restart ..." ...
```

- Remember to increase the value of `end_time` in the parameter file before submitting the job.
- If a manually started restart job shall continue a run of a former job chain which is somewhere in the middle of this chain, all binary files with respective higher cycle numbers have to be deleted or removed from their respective directories.