ForWind
Center for Wind Energy Research

Installation of PALM on the FLOW cluster at Carl von Ossietzky University of Oldenburg

Björn Witha

January 9th, 2013

Leibniz Universität Hannover
CARL VON OSSIETZKY universität OLDENBURG
Universität Bremen

Introductory comments

- Please have as well a look at the “official” PALM documentation, especially at the section “Installation”:

  http://palm.muk.uni-hannover.de/

- So please use both sources of information in parallel, when you are trying to install PALM on FLOW.
- There are two methods to install and run PALM:
  - The ***advanced method*** is the **standard method for installing and running PALM**. It allows using the full functionality of PALM. The installation procedure is using the script `mbuild`. All jobs are submitted using the script `mrun` (executed on FLOW within the SGE submission script).
  - The ***simple method*** is only using two scripts: `palm_simple_install` and `palm_simple_run`. It has several restrictions compared to the *advanced* (standard) *method* and is **only recommended for users who like to do a quick test of PALM!**
- You need a permit to access the PALM repository. If you have no permit so far, please contact the PALM group (raasch@muk.uni-hannover.de) and define a user name under which you like to access the repository. You will then receive a password which allows access under this name.
- In the following, first the standard method is described, afterwards the simple method (slide 20)

Installation of PALM *standard version* on FLOW (1)

In contrast to the normal application of PALM (jobs are submitted from a local host and run on a remote host), in this case PALM is directly installed and operated on the local host (FLOW). Hence, you don't need to establish a passwordless login as stated in the official PALM documentation. The scripts are already adapted for the application of PALM on FLOW (as of revision 755), so no modifications have to be made.

Login to FLOW and create the following directory, which will be called the working directory from now on (in principal, the directory name can be freely chosen, but it is recommended to chose “current_version” as a different name will require several modifications in the scripts):

```
/palm/current_version/
```

Change into the working directory and check out a working copy of the most recent PALM version (currently as of 09-Jan-2013 revision 1085):

```
cd ~/palm/current_version
svn checkout --username <your username>  /
        svn://130.75.105.7/palm/trunk  trunk
```

Replace <your username> with the name you have chosen to access the PALM repository (**not** your user name on FLOW!).

Installation of PALM *standard version* on FLOW (2)

The most recent version may still contain bugs. However, it is recommended as fixed releases are released only every 1-2 years.

At this time (09-Jan-2013), the most recent stable release is 3.9 (revision 1036).

You can download specific (older) releases by typing:

```
svn checkout --username <your username> \
     svn://130.75.105.7/palm/tags/release-<#>  trunk
```

where <#> has to be replaced by the any of the available PALM releases, e.g. "`3.9`".

To run PALM on FLOW, you should not use older releases than 3.9! Otherwise you would have to adjust the scripts manually!

You will be prompted for your password. After completion, a subdirectory `trunk` will appear in your working directory. It contains a number of further subdirectories, which contain e.g. the PALM source code (`SOURCE`) and the scripts for running PALM (`SCRIPTS`).

Installation of PALM *standard version* on FLOW (3)

To use the PALM scripts, the PATH-variable has to be extended and the environment variable `PALM_BIN` has to be set in your `.bashrc` file (located in your home directory – if not, create a new file starting with the line "`# .bashrc -l`") by adding the following two lines:

```
export PALM_BIN=$HOME/palm/current_version/trunk/SCRIPTS
export PATH=$PALM_BIN:$PATH
```

Before compiling the PALM code, several modules have to be loaded/unloaded in your `.bashrc` file as well:

```
module unload gcc
module load netcdf/4.2.1.1/intel/64/2011.0.013
module load netcdf-fortran/4.2/intel/64/2011.0.013
module load intel/ics/64/2011.0.013
module load intel/impi/4.0.1.007/64
module load hdf5/1.8.9/intel/64/2011.0.013
module load qt4/4.8.2
```

Please logout and then login again to activate these settings.

Installation of PALM *standard version* on FLOW (4)

To install and run PALM, a configuration file `.mrun.config` is required. There are some default versions in the directory `trunk/SCRIPTS`. To use PALM on FLOW please copy the following file to your working directory:

```
cd /palm/current_version
cp /trunk/SCRIPTS/.mrun.config.forwind  .mrun.config
```

The configuration file contains all information for compiling the PALM code and for generating and running the PALM executable using MPI. You will have to edit this file as specified below.

The first block of .mrun.config specifies several paths used below to specify where the input and output data is stored. In principle, you are free to chose the paths. However, as the available space in the $HOME directory of FLOW is only 50 GB per user, it is strongly recommended to store the output data in your individual data directory `/data/work/fw/<user-id>/` where `<user-id>` represents your user name on FLOW.

Installation of PALM *standard version* on FLOW (5)

The following settings are recommended (please replace `<user-id>` with your username on FLOW!):

```
#------------------------------------------------------------------------------
%mainprog          palm.f90
%base_directory    $HOME/palm/current_version
%base_data         $base_directory/JOBS
%output_data       /data/work/fw/<replace by your username>/palm/current_version/JOBS
# Replace <replace by your username> by your username on FLOW. Replace "/fw/" by "/iwes/" if
# your user directory resides in /iwes/.
%source_path       $base_directory/trunk/SOURCE
%add_source_path   $base_directory/USER_CODE/$fname
%depository_path   $base_directory/MAKE_DEPOSITORY
%use_makefile      true
#
```

In the second block the host identifier „lcflow“ is assigned to the hostnames `flow*` (the login nodes, i.e. “flow01”), `cfdl*` (the low-memory computing nodes, i.e. “cfdl001”) and `cfdh*` (the high-memory computing nodes).

```
#
%host_identifier       flow*           lcflow
%host_identifier       cfdl*           lcflow
%host_identifier       cfdh*           lcflow
%host_identifier       cfdx*           lcflow
#
```

Installation of PALM *standard version* on FLOW (6)

The next block contains information for compiling the PALM code: compilers, compiler options, netCDF libraries,...

Please replace `<user-id>` with your username on FLOW everywhere. By default, the temporary files during a job are stored in a directory `tmp` and the restart data is stored in a directory `palm_restart_data`, both in the data directory `/data/work/fw/<user-id>/`.

With the specified compiler, compiler options and netCDF libraries PALM is currently running well. However, the model performance may be optimized using different options or compilers in future.

```
# The next block contains all informations for compiling the PALM code
# and for generating and running the PALM executable using MPI. Replace <user-id>
# with your username on FLOW and if necessary "/fw/" by "/iwes/".
# The following 'lcflow parallel' block assumes PALM compilation and execution
# on host flow. Please note that other hosts may require different settings.
#
%remote_username   <replace by your username>                                    lcflow parallel
%tmp_user_catalog  /data/work/fw/<replace by your username>/tmp                  lcflow parallel
%tmp_data_catalog  /data/work/fw/<replace by your username>/palm_restart_data    lcflow parallel
%compiler_name     mpiifort                                                      lcflow parallel
%compiler_name_ser ifort                                                         lcflow parallel
%cpp_options       -cpp:-DMPI_REAL=MPI_DOUBLE_PRECISION:-DMPI_2REAL=MPI_2DOUBLE_PRECISION:-D__netcdf:-D__netcdf4:-D__lc:-D__parallel     lcflow parallel
%netcdf_inc        -I$$NETCDFF_INCLUDE                                           lcflow parallel
%netcdf_lib        -L$$NETCDFF_DIR:-lnetcdff:-L$$NETCDF_DIR:-lnetcdf             lcflow parallel
%mopts             -j:4                                                          lcflow parallel
%fopts             -xSSE4.2:-O3:-r8:-align:all:-ftz:-fno-alias:-no-scalar-rep:-ip:-nbs:-convert:little_endian              lcflow parallel
%lopts             -xSSE4.2:-O3:-r8:-align:all:-ftz:-fno-alias:-no-scalar-rep:-ip:-nbs:-Vaxlib                             lcflow parallel
%login_init_cmd    .:/cm/local/apps/environment-modules/3.2.6/Modules/3.2.6/init/bash\;:MODULEPATH=/cm/shared/modulefiles/   lcflow parallel
%modules           netcdf/4.2.1.1/intel/64/2011.0.013:netcdf-fortran/4.2/intel/64/2011.0.013:intel/ics/64/2011.0.013:intel/impi/4.0.1.007/64:hdf5/1.8.9/
intel/64/2011.0.013:qt4/4.8.2  lcflow parallel
#
```

Installation of PALM *standard version* on FLOW (7)

As a next step, the script mbuild has to be run to generate the executables:

```
mbuild -u -h lcflow
```

Please answer all queries with 'y'. If the executables have been generated successfully, the message "*** mbuild finished" will be displayed.

To avoid the re-compilation of the complete source code for each model run, PALM will be pre-compiled once by again using the script `mbuild`. The subroutines have to be compiled in a certain order. Therefore the so-called make mechanism is used, requiring a Makefile, in which the dependencies are described. This file is found in the subdirectory `trunk/SOURCE` where also the PALM code is stored. The compiled sources (object file `*.o`) are stored in the default directory `palm/current_version/MAKE_DEPOSITORY_parallel`. Please type:

```
mbuild -h lcflow
```

Again, all queries have to be answered with 'y' (the compilation is done for each block in your .mrun.config separately. By default, there is only one block: 'lcflow parallel').
The compilation may take several minutes. The compiled routines will appear in the shell. Usually some warning or information messages appear (like: *"warning #5462: Global name too long"* or *"remark #8290: Recommended relationship between field width 'W' and the number of fractional digits 'D' in this edit descriptor is ...")* but hopefully no error messages.
If the compilation has been successful, „*** mbuild finished" will be displayed.

Installation of PALM *standard version* on FLOW (8)

After having compiled PALM, the following lines in your `.bashrc`, which you added in order to install PALM, have to be removed or commented out (they will instead be included in the SGE submission script):

```
#export PALM_BIN=$HOME/palm/current_version/trunk/SCRIPTS
#export PATH=$PALM_BIN:$PATH

#module unload gcc
#module load netcdf/4.2.1.1/intel/64/2011.0.013
#module load netcdf-fortran/4.2/intel/64/2011.0.013
#module load intel/ics/64/2011.0.013
#module load intel/impi/4.0.1.007/64
#module load hdf5/1.8.9/intel/64/2011.0.013
#module load qt4/4.8.2
```

To activate the settings, you have to logout and then login again.

Installation of PALM *standard version* on FLOW (9)

As a last step, after the compilation has been finished, the PALM installation has to be verified. For this purpose, a simple test run is carried out. Please create a job directory for the example run and copy the parameter file to this directory:

```
cd palm/current_version
mkdir -p JOBS/example_cbl/INPUT
cp trunk/INSTALL/example_cbl_p3d \
                  JOBS/example_cbl/INPUT/example_cbl_p3d
```

To run PALM on FLOW, you need a job submission script, which contains options and configurations for the SGE queueing system as well as necessary options and commands to run PALM. A sample SGE script for PALM jobs can be found in the HPC User Wiki:

http://wiki.hpcuser.uni-oldenburg.de/index.php/Palm.sge

Please copy the sample script to your working directory (as `palm.sge`). For carrying out the test run, the script does not need to be modified. For different runs, you have to modify the script which is described in the following.

Installation of PALM *standard version* on FLOW (10)

Besides specifying the Korn-shell to be used, the section „General options“ contains also the name of the SGE-job (can be freely selected – default is 'palm').

Again, the `PATH`-variable has to be extended and `PALM_BIN` to be set, as we are now in the Korn-shell and `.bashrc` is no longer valid.

```
####################
# General options #
####################

### Write stdout and stderr in only one file: ###
#$ -j y

### PALM requires the Korn-shell: ###
#$ -S /bin/ksh

### name of the job ###
#$ -N palm

### change to directory where job was submitted from: ###
#$ -cwd

### extend PATH-variable to use PALM scripts ###
### and set environment variable PALM_BIN    ###
export PALM_BIN=$HOME/palm/current_version/trunk/SCRIPTS
export PATH=$PALM_BIN:$PATH
```

Installation of PALM *standard version* on FLOW (11)

The section „Resource requirements of the job“ contains specifications like

- Maximum running time of the job
- required memory
- switch to high-memory nodes
- required disk space

Please see http://wiki.hpcuser.uni-oldenburg.de for further explanations!

```
#######################################
# Resource requirements of the job #
#######################################

### maximum walltime of the job (hh:mm:ss): ###
#$ -l h_rt=0:10:0

### memory per job slot (maximum: 1.9G for low-memory nodes, ###
### 3.8G for high-memory nodes):                             ###
#$ -l h_vmem=100M

### uncomment line if high-memory node is needed (>23 GB per node): ###
##$ -l highmem=true

### disk space - default: 50 GB (if too low, job aborts!)    ###
### if job generates data larger than 50G --> increase value ###
#$ -l h_fsize=50G
```

Installation of PALM *standard version* on FLOW (12)

For parallel jobs like the PALM runs, additionally the number of slots has to be specified. The test case (a very small, undemanding job) should be carried out on 8 slots.

It is recommended to enable resource reservation for parallel jobs – this guarantees that resources a dedicated to jobs in job-priority order and prevents that a lot of small lower priority jobs block the requested nodes. Resources are guaranteed to be available as soon as possible.

```
#########################################
# Specific options for parallel jobs #
#########################################

### which parallel environment to use, and number of slots ###
#$ -pe impi 8

### enable resource reservation (to prevent starving of parallel jobs) ###
#$ -R y
```

Installation of PALM *standard version* on FLOW (13)

In this section, specifications needed for running PALM are made:

- The required modules have to be loaded, conflicting modules unloaded
- The execution script `mrun` has to be called (see next slide for options)

```
#------------------------------------------------------------------------------#
#                         Commands to be executed by the job                   #
#------------------------------------------------------------------------------#

echo `date`
echo

### it's useful (and nice) to see on which machines the job is running ###
echo "Contents of PE_HOSTFILE:"
cat $PE_HOSTFILE
echo

### load required modules / unload conflicting modules ###
module unload gcc
module load netcdf/4.2.1.1/intel/64/2011.0.013
module load netcdf-fortran/4.2/intel/64/2011.0.013
module load intel/ics/64/2011.0.013
module load intel/impi/4.0.1.007/64
module load hdf5/1.8.9/intel/64/2011.0.013
module load qt4/4.8.2

### call script which starts palm ###
mrun -z -d example_cbl -h lcflow -K parallel -X 8 -t 600 -r "d3# pr#"

exit
```

Installation of PALM *standard version* on FLOW (14)

```
### call script which starts palm ###
mrun -z -d example_cbl -h lcflow -K parallel -X 8 -t 600 -r "d3# pr#"
```

The most common `mrun` options which should be specified for every job are:

-d <job name>	name of the job (must be equal to directory name in `JOBS/`)
-h <host name>	host name – for jobs on FLOW always 'lcflow'
-K <block descriptor>	block descriptor in .mrun.config, by default 'parallel'
-X <number of slots>	number of required slots (must be consistent to the number of slots specified above!)
-t <simulation time>	required simulation time (should be consistent to the time specified above by parameter `h_rt` but in seconds)
-z	suppresses namelist file check - currently required for simulations on FLOW as namelist file check does not work properly on FLOW
-r <file connection statements>	desired output files d3# is required for all jobs! pr# → output of profiles ts# → output of time-series 3d# → output of 3d data Further output types are listed in `.mrun.config` Add 'restart' if you want to output restart data, needed for restart runs

Installation of PALM *standard version* on FLOW (15)

Submit the example run by simply typing:

```
qsub palm.sge
```

All jobs are submitted with this command.

By typing `qstat` you can check if your job is actually running. If the job does not appear in the list, it has already finished (the test run will be finished in much less than a minute).

In the working directory should now appear 2 files (* stands for the job number):

```
palm.o*          palm.po*
```

Please check them carefully, especially if problems have occurred. Especially important is the `.o*` file – it contains the job protocol with (in case of aborts) error messages.

The error message (“*/etc/profile[61]: .[6]: .[6]: .[40]: shopt: not found [No such file or directory]*”), can be ignored, as it does not affect the PALM runs.

Installation of PALM *standard version* on FLOW (16)

Besides the 2 output files in the working directory, 3 monitoring files should be found in:

```
~/palm/current_version/JOBS/example_cbl/MONITORING
```

`example_cbl_cpu`	contains the cpu time measures
`example_cbl_header`	contains general information about the run and the selected parameters
`example_cbl_rc`	contains the content of `example_cbl_header` plus the values of certain control parameters at certain temporal intervals

The netCDF output files are stored by default in the directory:

```
/data/work/fw/<user-id>/palm/current_version/JOBS/example_cbl/OUTPUT
```

To verify the results of this example run, please compare the content of the run control file with the example result file:

```
diff JOBS/example_cbl/MONITORING/example_cbl_rc \
                 trunk/INSTALL/example_cbl_rc
```

You should not find any difference between these two files, except for the run date and time displayed at the top of the file header and, maybe, the number of cores that have been used.

If the file contents are identical, the installation is successfully completed!

Your own PALM runs on FLOW

To carry out your own PALM runs, do the following:

- Create a new folder in `JOBS/` with the desired name of your run
- In this folder create a folder `INPUT/` and copy the parameter file from the test case there (`example_cbl_p3d`)
- Change its name (it has to be consistent with the job folders name!) and modify the `p3d`-file according to your requirements
- Modify the SGE submission script `palm.sge` (job name, cores, simulation time, memory, etc.) or create a new script
- Submit the job with `qsub`
- Monitor the job with `qstat, qstat -j <job number>, qhost, qhost -j`
- Temporary RUN_CONTROL (during the run) can be found in `/data/work/fw/<user-id>/tmp/`
- When the job has finished, check the respective palm.o* file (also possible during the run)
- Check the files in `JOBS/<job name>/MONITORING`
- Output is stored in `/data/work/fw/<user-id>/palm/current_version/JOBS/<job name>/OUTPUT`

Installation of PALM *simple version* on FLOW (1)

Login to FLOW and create the following directory, which will be called the working directory from now on (in principal, the directory name can be freely chosen):

```
/palm/simple_version/
```

Change into the working directory and check out a working copy of the most recent PALM version:

```
cd ~/palm/simple_version
svn checkout --username <your username>  /
        svn://130.75.105.7/palm/trunk  trunk
```

Replace <your username> with the name you have chosen to access the PALM repository.

The most recent version may still contain bugs. However, it is recommended as fixed releases are released only every 1-2 years.

For checking out the most recent stable release or older versions see slide 4.

You will be prompted for your password. After completion, a subdirectory `trunk` will appear in your working directory. It contains a number of further subdirectories, which contain e.g. the PALM source code (`SOURCE`) and the scripts for running PALM (`SCRIPTS`).

Installation of PALM *simple version* on FLOW (2)

Extend the PATH-variable to use PALM and set environment variable `PALM_BIN` in your `.bashrc` file (located in your home directory – if not, create a new file starting with the line "`# .bashrc -1`") by adding the following two lines:

```
export PALM_BIN=$HOME/palm/simple_version/trunk/SCRIPTS
export PATH=$PALM_BIN:$PATH
```

Before compiling the PALM code, you have to load/unload several required modules in your `.bashrc` file:

```
module unload gcc
module load netcdf/4.2.1.1/intel/64/2011.0.013
module load netcdf-fortran/4.2/intel/64/2011.0.013
module load intel/ics/64/2011.0.013
module load intel/impi/4.0.1.007/64
module load hdf5/1.8.9/intel/64/2011.0.013
module load qt4/4.8.2
```

You may logout and then login again to activate these settings.

Installation of PALM *simple version* on FLOW (3)

Call the installation script:

```
cd /palm/simple_version
palm_simple_install -i MAKE.inc.ifort.forwind
```

The script copies the PALM source code into a new subdirectory `MAKE_DEPOSITORY_simple`. This directory also will contain a `Makefile` and an include file `MAKE.inc` to be used for compiling the code in the next step.

It also creates a directory `JOBS/example_cbl` which is needed to carry out a test run after the installation has been finished.

The include file `MAKE.inc` (in the just created directory `MAKE_DEPOSITORY_simple`) contains compiler name, compiler options, library path for netCDF, etc.. It is already adjusted to FLOW.

Installation of PALM *simple version* on FLOW (4)

To compile the PALM code, change to the `MAKE-depository` and execute make:

```
cd palm/simple_version/MAKE_DEPOSITORY_simple
make
cd ..
```

The compilation will take some minutes. The compiled routines will appear in the shell. Usually some warning or information messages appear (like: *“warning #5462: Global name too long”* or *“remark #8290: Recommended relationship between field width 'W' and the number of fractional digits 'D' in this edit descriptor is ...”)* but hopefully no error messages.

After having compiled PALM, the following lines in your `.bashrc`, which you added in order to install PALM, have to be removed or commented out (they will instead be included in the SGE submission script):

```
#export PALM_BIN=$HOME/palm/simple_version/trunk/SCRIPTS
#export PATH=$PALM_BIN:$PATH

#module unload gcc
#module load netcdf/4.2.1.1/intel/64/2011.0.013
#module load netcdf-fortran/4.2/intel/64/2011.0.013
#module load intel/ics/64/2011.0.013
#module load intel/impi/4.0.1.007/64
#module load hdf5/1.8.9/intel/64/2011.0.013
#module load qt4/4.8.2
```

Installation of PALM *simple version* on FLOW (5)

PALM is executed by a script called `palm_simple_run` which is located in:

```
trunk/SCRIPTS/
```

Please activate the appropriate `mpiexec`-command for runs on FLOW near the end of the script palm_simple_run and deactivate the command for runs at IMUK:

```
# IMUK:
# mpiexec  -machinefile hostfile  -n $mpi_procs  ./palm  < runfile_atmos

# SGI-MPT HLRN:
# mpiexec_mpt  -np $mpi_procs  ./palm  < runfile_atmos

# HPC-FLOW (ForWind):
 mpiexec  -machinefile $TMPDIR/machines -n $mpi_procs  -env I_MPI_FABRICS shm:ofa ./palm  < runfile_atmos
```

Installation of PALM *simple version* on FLOW (6)

After the installation and compilation has been finished, a test run should be carried out in order to check the installation.

To run PALM on FLOW, you need a job submission script, which contains options and configurations for the SGE queueing system as well as necessary options and commands to run PALM. A sample SGE script for PALM jobs can be found here:

```
/user/fw/zugo5243/Examples/palm_simple.sge
```

Please copy the sample script to your working directory.

To activate the settings, you have to logout and then login again.

Installation of PALM *simple version* on FLOW (7)

Most of the script is similar to the SGE script for the standard version of PALM – see slides 12 – 15 for explanations. The only significant difference is the call of the script which runs PALM, in this case `palm_simple_run` (at the end of the script).

```
### call script which starts palm ###
palm_simple_run -p 4 -c example_cbl

exit
```

It has the following options:

- Option **`-p`** specifies the number of slots/cores needed for the job – please make sure, that it is consistent to the number specified in the previous section!
- Option **`-c`** specifies the actual name of the PALM run to be executed – in this case the test run named `example_cbl`

Installation of PALM *simple version* on FLOW (8)

The test run (as every PALM run) requires a parameter file for steering PALM, which is in FORTRAN-NAMELIST format. This file has been already generated by the installation script `palm_simple_install` under `JOBS/example_cbl/INPUT/example_cbl_p3d`.

PALM is started by submitting the SGE script `palm_simple.sge` with `qsub` (from your working directory):

```
qsub palm_simple.sge
```

By typing `qstat` you can check if your job is actually running. If the job does not appear in the list, it has already finished (the test run will be finished in much less than a minute).

In the working directory should now appear 2 files (* stands for the job number):

```
palm_simple.o*          palm_simple.po*
```

These files contain the standard output or error messages. Please check them carefully, especially if problems have occurred.

The error message (“*/etc/profile[61]: .[6]: .[6]: .[40]: shopt: not found [No such file or directory]*”), can be ignored, as it does not affect the PALM runs.

Installation of PALM *simple version* on FLOW (9)

In addition to the four output files, each run creates an output directory named *OUTPUT.<job name>+<number of cores>+<submission date and time>*, e.g.:

```
OUTPUT.example_cbl+4+12.Jul_14:59:00/
```

This directory contains several files:

```
[zugo5243@flow01 OUTPUT.example_cbl+4+12.Jul_14:59:00]$ ls -al
total 9798
drwxr-xr-x 2 zugo5243 fw     499 Jul 12 14:59 .
drwxr-xr-x 6 zugo5243 fw     936 Jul 12 14:59 ..
-rw-r--r-- 1 zugo5243 fw    3038 Jul 12 14:59 CPU_MEASURES
-rw-r--r-- 1 zugo5243 fw    9184 Jul 12 14:59 DATA_1D_PR_NETCDF
-rw-r--r-- 1 zugo5243 fw   29088 Jul 12 14:59 DATA_1D_TS_NETCDF
-rw-r--r-- 1 zugo5243 fw  110128 Jul 12 14:59 DATA_2D_XY_NETCDF
-rw-r--r-- 1 zugo5243 fw   16128 Jul 12 14:59 DATA_2D_XZ_AV_NETCDF
-rw-r--r-- 1 zugo5243 fw   57536 Jul 12 14:59 DATA_2D_XZ_NETCDF
-rw-r--r-- 1 zugo5243 fw      33 Jul 12 14:59 DEBUG_0000
-rw-r--r-- 1 zugo5243 fw      33 Jul 12 14:59 DEBUG_0001
-rw-r--r-- 1 zugo5243 fw      33 Jul 12 14:59 DEBUG_0002
-rw-r--r-- 1 zugo5243 fw      33 Jul 12 14:59 DEBUG_0003
-rw-r--r-- 1 zugo5243 fw     235 Jul 12 14:59 ENVPAR
-rw-r--r-- 1 zugo5243 fw    6340 Jul 12 14:59 HEADER
-rw-r--r-- 1 zugo5243 fw      32 Jul 12 14:59 hostfile
-rwxr-xr-x 1 zugo5243 fw 6892605 Jul 12 14:59 palm
-rw-r--r-- 1 zugo5243 fw    1081 Jul 12 14:59 PARIN
-rw-r--r-- 1 zugo5243 fw   59261 Jul 12 14:59 RUN_CONTROL
-rw-r--r-- 1 zugo5243 fw      12 Jul 12 14:59 runfile_atmos
[zugo5243@flow01 OUTPUT.example_cbl+4+12.Jul_14:59:00]$
```

CPU measures
CPU time, the run needed (in total and for several components of the model)

netCDF output data
(in this case profiles, timeseries, instantaneous xy-cross sections, instantaneous and time-averaged xz-cross sections)

Run control
You can look in this file already during the run to check if everything is ok and to extrapolate the running time. It contains also values of some important variables.

Installation of PALM *simple version* on FLOW (10)

To verify the results of this example run, compare it with the default result:

```
diff OUTPUT.example_cbl*/RUN_CONTROL  trunk/INSTALL/example_cbl_rc
```

You should not find any difference between these two files, except for the run date and time displayed at the top of the file header and, maybe, the number of cores that have been used.

If the file contents are identical, the installation is successfully completed!