# Submitting FLUENT6 and CFX5 Jobs to the Center for Parallel Computers (PDC), KTH

Pedro Costa
Stockholm, January 2006
pedro.costa@energy.kth.se

# Introductory Note

It is expected that the reader is familiar with the basics of the EASY Scheduler (queue system on PDC), Kerberos (telnet client) and has read through necessary documentation available on the PDC web site, namely the "Guided Tours" http://www.pdc.kth.se/support/tours.html
Basic UNIX and shell scripting knowledge is required.
The procedures described in this report are subject to modifications. The information provided here is relative to CFX 5.7.1 and Fluent 6.1.22 and the current environment setup at PDC. The *Lucidor* cluster `blumino.pdc.kth.se` was used in all simulations.

This report is a **first step** in helping those who will be running CFD programs on PDC. The procedures described in this report are flexible and different submitting procedures can be considered. Contributions to this document and to the scripts presented here are more than welcome and can be sent to the author's address.

A few minor corrections/additions have been made by PDC staff members ulfa and haba. The original front page was lost when converting from Word to OpenOffice.

Happy computing!

# Table of Contents

# I. PDC[1]

The Center for Parallel Computers operates leading-edge, high-performance computers as easily accessible national resources. These resources are primarily available for Swedish academic research and education. A couple of the major computers available at PDC are presented in the following table.

| Name | Type | Description |
|---|---|---|
| *Lenngren* | Intel Xeon Cluster | A 442 node Intel Xeon cluster. |
| *Lucidor* | HP Itanium2 Cluster | A 90 node HP Itanium2 cluster with 180 64-bit processors. The processors are running at 900MHz giving a peak performance of 7.2GFlop/s per node (2 CPUs per node and 4 flops per clock cycle). Each node has 6Gbyte memory. |

Most of the procedures described in this report are relative to the Itanium Processor Family, cluster *Lucidor* at PDC, but they are easily extended to other systems running Fluent and CFX. Before running a program on *Lucidor* you will have to allocate a "resource" in the system to run on, *i.e.,* you will need to "book" one or more nodes on the system to run on. You can book two types of resources:

**Interactive nodes**

Interactive nodes are shared pretty much in the same way as the login-node is. This means that at any one moment more than one user is running on each interactive node and also when you allocate several interactive nodes to run a parallel program you will find that you are running several instances of your job on each interactive node, *i.e,.* the interactive nodes are used as pseudo-nodes. A five way parallel job when run interactively may actually run as 3 instances on interactive node no. 1 and two instances on interactive node no. 2. *Interactive nodes are intended for short tests.*

**Dedicated nodes**

Dedicated nodes or batch nodes are used as unique nodes in the parallel execution. On a dedicated node you are guaranteed to be the only user on the node. Commonly you will run one instance of your parallel program on each dedicated node you have allocated giving possibility to use 100% of the computing power of each node to your program.

*In this report, only the procedures relative to the dedicated nodes will be described.*

---

[1] information extracted from PDC website http://www.pdc.kth.se

# I.1 Connecting to PDC

Log on to whatever cluster you are using on PDC, for example *Lucidor* where the login node is called `blumino.pdc.kth.se`. Log in with rxtelnet from your workstation according to the instructions in the Kerberos Tour on the PDC webpages. Assuming you are using the bash shell, in your home directory, alter your `.bashrc` file with the `-vi` editor, so that the necessary modules will always be loaded. A sample of the `.bashrc` file:

```
# ------------------------------------------------------------------------------------------
# Add necessary modules for running CFX and Fluent on Lucidor:
# ------------------------------------------------------------------------------------------
module add heimdal
module add X11/R6
module add easy
module add fluent
module add cfx
module add i-compilers/7.1-47
module add mpich
# ------------------------------------------------------------------------------------------
```

Note that the i-compilers version used with CFX has to be the one indicated in this list. The most recent i-complier version (default) did not work properly at the time of this report. This is probably because 7.1-47 was the default compiler when cfx was installed (ulfa, PDC).

If you do not have a `.bashrc` file, create one in your Public directory and do

```
cd .
ln –s Public/.bashrc .
```

Note that your login files must not contain stuff that causes error messages. Putting the stuff above in your `.bashrc` will cause failure when submitting batch jobs on Lenngren. (ulfa, PDC staff)

# II. FLUENT 6

## Description

Fluent is a leading computer program for modeling fluid flow and heat transfer in complex geometries and is a Finite Volume based code. Fluent provides complete mesh flexibility, solving flow problems with unstructured meshes that can be generated about complex geometries with relative ease.

Fluent can be used for:

- analysis of incompressible or compressible, steady-state or transient, inviscid, laminar, and turbulent flows
- flows with Newtonian as well as non-Newtonian behavior
- analysis of convective (natural and forced), coupled conduction/convective and radiation heat transfer
- simulations using multiple moving reference frames, including sliding mesh interfaces and mixing planes for rotor/stator interaction modeling
- flows chemical species mixing and reaction, including combustion sub models and surface deposition reaction models
- flows with arbitrary volumetric sources of heat, mass, momentum, turbulence, and chemical species
- Lagrangian trajectory calculations for a dispersed phase of particles/droplets/bubbles, including coupling with the continuous phase (drag, lift-forces on particles)
- flow through porous media
- two-phase flows, including cavitation
- free-surface flows with complex surface shapes
- one-dimensional fan/heat-exchanger performance simulations

In order to solve the CFD problem in parallel, the grid generated is partitioned and distributed among a set of tasks started on a given set of processors. Then during the solution phase, these processes perform iterative computation and cooperate with each other in arriving at the final solution. Check out http://www.fluent.com/about/news/newsletters/03v12i2_fall/pdfs/nl41.pdf for more information.

# Procedure

The steps involved in setting up a Fluent run are very similar to those that will be presented for CFX. Typically, you can read your mesh (generated in ICEM, for example) into Fluent on your local machine and setup all the necessary parameters interactively (directly using the GUI). You can then save this as a case file (`something.cas.gz`), FTP it to PDC and then prepare the journal file using a text editor on PDC (`-vi` , `-emacs` or `-nedit` are examples of text/script editors you can use). A small script must also be written to deal with the parallelization of Fluent on PDC.

Since the case file is compressed, it occupies less disk space on PDC (an important issue) and several parameters have already been setup in this file (the boundary conditions, basic discretization schemes, fluid and turbulence models, etc.). This strategy greatly simplifies the writing of the journal file and avoids syntax errors.

# II.1 Fluent Journal File

The journal file is the equivalent of the definition file in CFX. It is written in a dialect of Lisp called Scheme and contains all the instructions that are to be executed during the run. A basic form of this file, using the `-vi` editor, is as follows:

```
# ----------------------------------------------------------------------------
# SAMPLE JOURNAL FILE
#
# read case file (*.cas.gz) that had previously been prepared
file/read-case something.cas.gz

# initialize flowfield
solve/initialize/initialize-flow

# run 100 iterations
solve/iterate 100

# write data
file/write-data something.dat.gz

# exit fluent
exit yes
#-----------------------------------------------------------------------------
```

Save the journal file as `something.jou`.

In this simple example, we read in the `something.cas.gz` file, which we had previously prepared (includes all the boundary conditions and models). We then initialized the flowfield in order to start the iterations and we solved for 100 iterations. After 100 iterations had been performed, the data generated was written to the file `something.dat.gz`. We are assuming here that 100 iterations will suffice for a converged solution.

## II.1.1 Convergence Issues

Up to here everything seems quite straightforward. However, imagine that during the run the solution diverged, i.e., it would arrive at iteration 100 and we would not have a converged solution or, the simulation would crash sometime in between. There are several ways to perform a more stable simulation. For example, start off with simpler turbulence models (or/and fluid models or/and more relaxed discretization schemes) and perform *n* number of iterations. Then change to more elaborate models a perform *m* iterations.

An example of this strategy would look something like this (we assume here that `something.cas.gz` had been setup with the "simple" standard k-epsilon turbulence model):

```
# ------------------------------------------------------------------------------
# SAMPLE JOURNAL FILE (attempt to avoid some convergence issues)

# read case file (*.cas.gz) that had previously been prepared with standard # k-epsilon
turbulence model
file/read-case something.cas.gz

solve/initialize/initialize-flow

solve/iterate 50
file/write-data something-50.dat.gz

# change to a more elaborate turbulence model
define/model/viscous/ke-realizable yes

# run another 50 iterations
solve/iterate 50

# write case and data files to current directory
file/write-case-data something-100.dat.gz

# exit fluent
exit yes
#------------------------------------------------------------------------------
```

Check Fluent documentation for further information on convergence strategies.

## II.1.2 Fluent Text Inputs for Journal Files (Scheme)

If you are not familiar with the nomenclature of the Fluent text inputs, do the following:
Open the GUI version of Fluent. Once loaded, press <enter> and you will see the text version of the "main menu". Type `file` and press <enter>. Type `read-case` and press <enter>. You get the picture…Type `q` and press <enter> to exit the submenus.
In a journal file, this would look something like this:

```
> file/read-case/
```

*or (in condensed notation)*

```
> f/rc/
```

**Warning!**
Careful when using the condensed notation. There may be several options for the same condensed notation, which makes it sometimes rather umbiguous. Test your journal files out before submitting your jobs to PDC.

Furthermore, the scheme language is programmable. An experienced user can easily automate several functionalities within the journal file.

# II.2 Running FLUENT on Dedicated Nodes–Batch Run (`esubmit`)

Only the submission process for batch runs on dedicated nodes will be described here (for CFX, the interactive *use* of dedicated nodes will also be described - `spattach`). Do not confuse interactive *nodes* with interactive *use* of dedicated nodes, as I did for some time!

**Important**
When referring to interactive *nodes*, we are submitting the jobs via `spattach –i`, meaning that at any one moment more than one user can be running a program on the same nodes (shared `spattach`). When referring to interactive *use* of dedicated nodes, we are submitting our jobs to dedicated nodes via the `esubmit` command (and thus we are the only users of the nodes) and then running our program via `spattach` (non-interactive `spattach`).
Note that PDC clusters have a variable amount of interactive and dedicated nodes. Take care not to spawn large jobs on the interactive nodes, and to remove processes should your execution terminate not-normally. They should be used only to confirm if your submission process has been set up appropriately.

Connect to whatever cluster you will be using on PDC, using Kerberos telnet. Load the necessary modules. Once you have your journal file prepared, you will need to prepare a script to parallelize your Fluent run. Let us first look at the syntax involved in submitting a Fluent parallel simulation from the command line:

```
fluent <version> –g –p<comm> –t<N> –cnf=<host list> –i <journal file>
```

| | Description | Options/ Example | Notes |
|---|---|---|---|
| **<version>** | Solver version | **2d**<br>**2ddp**<br>**3d**<br>**3ddp** | 2 dimensional solver<br>2 dimensional double precision solver<br>3 dimensional solver<br>3 dimensional double precision solver |
| **<comm>** | Network communicator used to pass data among processors | **net**<br>**nmpi**<br>**smpi**<br>**vmpi** | Socket communicator<br>MPICH (distributed memory MPI)<br>MPICH (shared memory MPI only)<br>Vendor MPI |
| **<N>** | Number of CPU's to be used (not the number of nodes). Note that 1 node may contain several processors. | **---** | Depends on cluster and on the number of nodes that have been allocated to you once you have submitted a job. A script must be written to deal with this. For example, *Lucidor* blumino.pdc.kth.se has 2 CPU's per node, and 90 nodes = 180 CPU's. If you were to submit a job on 8 nodes and wanted to take advantage of the 2 processors per node, the variable N would be 2 x 8 =16 |
| **<host list>** | text file containing list of hosts | **something.hosts** | Care must be taken to format this file properly when submitting Fluent jobs. For example, if you want to run a Fluent job on 4 nodes using only 1 processor per node, this file would have to look something like this:<br><br>node1.pdc.kth.se:1<br>node2.pdc.kth.se:1<br>node3.pdc.kth.se:1<br>node4.pdc.kth.se:1<br><br>However, if you want to take advantage of the 2 processors per node, then the host list must look like this:<br><br>node1.pdc.kth.se:2<br>node2.pdc.kth.se:2<br>node3.pdc.kth.se:2<br>node4.pdc.kth.se:2<br><br>The script presented in the following section deals with getting this syntax correct, so don't worry…. |
| **<journal file>** | text file that contains the instructions for the Fluent run | **something.jou** | Dealt with in section II.1 |

At this point, lets have a look at some questions that may be bothering you:

**Now that I know the syntax involved in submitting a parallel fluent job from the command line, how do I know which hosts will be allocated to me?**

The EASY scheduler system takes care of this. Once you type the `esubmit` command, several environment variables are generated, specific to the job you just submitted. One of these variables, the `SP_HOSTLIST`, contains a list of the hosts that have been allocated to you.
Lets suppose that you have requested 4 nodes `esubmit` –n4 …. then the `SP_HOSTLIST` would look something like this **(one node per line):**

```
node1.pdc.kth.se
node2.pdc.kth.se
node3.pdc.kth.se
node4.pdc.kth.se
```

**OK, so now I know which nodes will be allocated to me (`SP_HOSTLIST`). However, I've noticed that I must put this file in the correct format for Fluent, depending on how many CPUs per node I want to use. How do I do this?**

This is where your shell scripting knowledge comes into scene. We know that if you request 4 nodes and you want to take advantage of the 2 CPU's per node, then your Fluent host file will have to look something like this:

```
node1.pdc.kth.se:2
node2.pdc.kth.se:2
node3.pdc.kth.se:2
node4.pdc.kth.se:2
```

For this case, the fluent argument `-t` must have the number 8 in front of it (you will be running 4 x 2 = 8 processes (or tasks). All these adjustments can be done by writing a shell script, presented in the next section. Basically, the script just generates the Fluent syntax with the arguments in the correct format.

## II.3 The `fluentpdc.esy` Script

A script has been prepared called `fluentpdc,esy` and presented on the following page. Note that this script does not include all the arguments that you can supply to fluent, but you can/should add/replace whatever you want. As it is, it will work fine for your simulations if you follow the notation properly. You can copy this script to your home directory on PDC and save it as `fluentpdc.esy`. Make sure its permissions are set to executable. The `fluentpdc.esy` script handles all the arguments for fluent and puts them in the correct format

So, in order to run a fluent parallel job on PDC, and once you have created your journal file, all you have to do is type the following:

```
> esubmit -n4 -t120 -c CAC fluentpdc.esy 2 3ddp /home/journalfiles/case1.jou
```

In this example, we requested **4** nodes for **120** minutes (`esubmit`). Then the lrt`.esy` script comes into action and we choose to take advantage of the **2** CPUs per node (that *Lucidor* has, for example) to run the fluent **3d double precision** solver. Instructions are given to the Fluent solver via the journal file /home/journalfiles/**case1.jou**
To confirm if Fluent is actually running on the requested nodes, log on to one of them using `rxtelnet` and type `top`. Check the list to see if the Fluent executable appears and the CPU usage. Type `q` to exit `top`.

Once the job has finished, you will receive an email with the output. If you wish, you can automatically generate output files in a certain location (if you are familiar with shell scripting this should be easy for you to incorporate into `fluentpdc.esy`).
A help menu has been incorporated into the script, so just type `fluentpdc.esy -h` to access it if you need to be remembered of something.

The script `fluentpdc.esy` is available on Lucidor in /pdc/vol/fluent/6/Fluent.Inc/bin/fluent/

```
#!/bin/bash
# fluentpdc.esy script
# ---------------------- HELP MENU -----------------------------------------------
mail="pedro.costa@energy.kth.se"
help()
{
    cat <<HELP
fluentpdc.esy -- run fluent on a PDC cluster. This script must always be
                 preceeded by the esubmit script and its respective arguments.
USAGE:
       fluentpdc.esy [-h] -ppn -solver -journalfile

        Note that the order in which these arguments appear must be the one
       specified here

        -h             help text (this is an option)
       -ppn            number of processes to run on each node (1 or 2, for Lucidor)
       -solver         version of solver (2d, 2ddp, 3d, 3ddp)
       -journalfile    journal file (include path, if necessary)
EXAMPLE:
       esubmit -n4 -t120 -c CAC fluentpdc.esy 2 3ddp /home/journalfiles/case1.jou

       Here we requested 4 nodes for 120 minutes (esubmit). We chose to take
       advantage of 2 CPUs per node to run the fluent 3d double precision
       solver. Instructions will be given to the solver via the file
       /home/journalfiles/case1.jou
BUGS/SUGGESTIONS:
        $mail
HELP
    exit 0
}
# ------------------------------------------------------------------------------------
error()
{
    echo "$1"
    exit 1
}
# --------------------------- READ USER SUPPLIED OPTIONS ------------------------------
while [ -n "$1" ]; do
case $1 in
    -h) help;shift 1;;
    -*) echo "error: no such option $1. -h for help";exit 1;;
    *)  break;;
esac
done
PPN=$1;        # nr of processors per node
SOL=$2;        # version of solver
JOU="$3";      # path to journalfile
test $SP_PROCS || exit 255
# ------------------------ RUN FLUENT (PARALLELIZATION) -----------------------------
if [ $SP_PROCS -gt 1 ]; then
   #put SP_HOSTFILE in correct format and rename it
   sed 's/ *$/:'"$PPN/" $SP_HOSTFILE > $SP_JID.hosts
   #use distributed network MPI -- nmpi
   FLUENTARGS="-t`expr $PPN \* $SP_PROCS` -cnf=$SP_JID.hosts -pnmpi"
elif [ $PPN -gt 1 ]; then
   # use shared memory MPI only -- smpi
   FLUENTARGS="-t$PPN -psmpi"
fi
echo "Executing fluent in directory `pwd`"
echo "fluent $SOL -g $FLUENTARGS -i $JOU"
fluent $SOL -g $FLUENTARGS -i "$JOU"
```

## II.4 Fluent Summary

The steps required to submit a Fluent batch job on dedicated nodes to PDC are as follows:

1) Generate a mesh (in ICEM, for example). Save as `something.msh`
2) Open Fluent. Read in mesh and setup parameters (boundary conditions, models, discretization schemes, etc.). Save as `something.cas.gz` and FTP it to your working directory on PDC.
3) Login to whatever server you will be using on PDC. Confirm the lifetime of your tickets and increase it if necessary. Load necessary modules. Open the `-vi` editor and write your journal file. Save as `something.jou`
4) Copy the `fluentpdc.esy` script to your working directory. Make sure its permissions are set to executable and submit your job to PDC using the `esubmit` command.
5) Once the nodes requested have been allocated to you, confirm if everything is running properly by connecting to one of the nodes and typing `top`.
6) Once your convergence criteria has been met (or if the job crashes!) you will receive an email with the respective output

# III. CFX 5

## Description

CFX is a powerful finite-volume-based program package for modeling general fluid flow in complex geometries. The main components of the CFX package are the flow solver `cfx5solve`, the geometry and mesh generator `cfx5pre`, and the postprocessor `cfx5post`.
CFX implements a wide spectrum of physical models and numerical solution algorithms comparable to those of FLUENT.
Check out http://www.ansys.com for more information.

## III.1. Solver Definition File

If you have only a couple of runs to prepare, you can set them up in CFX-Pre interactively on your local machine (the other option would be batch – see CFX documentation for details).
Open CFX-Pre. Load mesh and setup solver definitions (boundary conditions, turbulence models, solver parameters, etc.). Save definition file for solver as `example.def`. The files generated with CFX-Pre (`example.cfx, example.gtm and example.def`) should then be FTP'd to your working directory at PDC.

## III.2 The `cfxpdc.esy` Script

The syntax involved in submitting a CFX parallel simulation from the command line can be found in the CFX documentation, namely the "Solver Manager" manual. The main difference compared to Fluent is the format of the host file, in which the hosts must be separated by commas. Also, when running more that 1 process per node, the hosts must appear multiplied by the number of processes. So again, we need to prepare a script to deal with this.

As previously mentioned you can submit your jobs on dedicated nodes for interactive use (`spattach`) or dedicated use (`esubmit`). The script `cfxpdc.esy` will deal with both situations. So lets have a look at it:

```
#!/bin/bash
# script cfxpdc.esy
# ------------------------ HELP INFORMATION -------------------------------------
mail="pedro.costa@energy.kth.se"
help()
{
    cat <<HELP
cfxpdc.esy -- run CFX on a PDC cluster. This script must always be
              preceeded by the esubmit script (dedicated use) or/and the
             spattach script (interactive use) and there respective arguments.
USAGE:
       cfxpdc.esy [-h] [-i] <ppn> <precision> <definitionfile>

        Note that the order in which these arguments appear must be the one
       specified here

        -h                    help text (this is an option)
       -i                     activate this option if you will be running this script
                              interactively, i.e., interactive use on dedicated nodes
                              (this is an option; default value is set to dedicated use
                              only)
       <ppn>                  number of processes to run on each node (1 or 2, for
                              Lucidor)
       <precision>            precision of solver [single, double]
       <definitionfile>       definitionfile (include path, if necessary)
EXAMPLE (interactive use):
       esubmit -n6 -t120 -T 2005-01-01/10:00:00
       rxtelnet node1.pdc.kth.se
       spattach -j <JID> -k cfxpdc.esy -i 2 double /home/definitionfiles/case1.def

       Here we requested, in advance, 6 nodes for 120 minutes (esubmit). We then
       logged on to one of the nodes (in this case node1.pdc.kth.se). We then
       chose to take advantage of 2 CPUs per node to interactively run the CFX double
       precision solver. Instructions will be given to the solver via the file /
       home/definitionfiles/case1.def
EXAMPLE (dedicated use):
       esubmit -n4 -t120 cfxpdc.esy 2 double /home/definitionfiles/case1.def

       Here we requested 4 nodes for 120 minutes (esubmit). We chose to take
       advantage of 2 CPUs per node to run the CFX 3d double precision
       solver. Instructions will be given to the solver via the file
       /home/defintionfiles/case1.def
BUGS/SUGGESTIONS:
        $mail
HELP
    exit 0
}
# -------------------------------------------------------------------------------
error()
{
    echo "$1"
    exit 1
}
# read OPTIONS and ARGUMENTS that user introduced --------------------------------
INT=0
while [ -n "$1" ]; do
case $1 in
    -h) help;shift 1;;
    -i) INT=1; shift 1;;
    -*) echo "error: no such option $1. -h for help";exit 1;;
    *)  break;;
esac
done
# ------------------------------------------------> script CONTINUES on next page ...
```

```
# ------------------------------------------------→ ... CONTINUATION of cfxpdc.esy
PPN=$1; # nr of processes per node
PRE=$2; # precision of solver
DEF=$3; # path of definition file

# ------------------ CFX for interactive use ----------------------------------
if [ $INT -eq 1 ]; then
   type_use="interactive"
   cat $SP_HOSTFILE | sort -u > i-hosts1.tmp
   # include the login node
   echo "ONLY VERY SHORT AND SMALL JOBS MAY BE ISSUED WHEN LOGIN NODE IS INCLUDED"
   echo "It is better to issue  spattach -I  on an interactive node"
   hostname >> i-hosts1.tmp
   # put SP_HOSTFILE in correct format (comma separated list)
   sed 's/ *$/*'"$PPN/" i-hosts1.tmp > i-hosts2.tmp
   awk '{ printf ",%s", $1; }' i-hosts2.tmp > i-hosts3.tmp
   cut -c 2- i-hosts3.tmp > $SP_JID.hosts

# ------------------ CFX for dedicated use ------------------------------------
elif [ $INT -eq 0 ]; then
   type_use="dedicated"
   # put SP_HOSTFILE in correct format (comma separated list)
   sed 's/ *$/*'"$PPN/" $SP_HOSTFILE > hosts1.tmp
   awk '{ printf ",%s", $1; }' hosts1.tmp > hosts2.tmp
   cut -c 2- hosts2.tmp > $SP_JID.hosts
fi

# note: pipes were not working properly, thus the excessive number
# of *.tmp files (these can all be deleted after simulation)


# ------------------ run CFX --------------------------------------------------
echo "Executing CFX - $type_use use in directory `pwd`"
echo "cfx5solve -def $DEF -$PRE -par-dist `cat $SP_JID.hosts`"
cfx5solve -def $DEF -$PRE -par-dist `cat $SP_JID.hosts`
# -----------------------------------------------------------------------------
```

Confused? Don't worry. For now, just save it to your working directory on PDC and remember to set it to executable. Read on and then come back to the script to see if you understand it better. A help menu has been created inside the script, so type `cfxpdc.esy -h` for help.

## III.3 Running CFX on Dedicated Nodes - Interactive Use (`spattach`)

Just follow these 3 steps. Reserve dedicated nodes in advance for interactive use. You are guaranteed to be the only user of the nodes:

```
> esubmit -n6 -t120 -c CAC -T 2005-01-01/10:00:00
```

Once you have been granted access to the nodes, logon to one of them:

```
> rxtelnet node1.pdc.kth.se
```

Now that you are on the node, run the `cfxpdc.esy` script as indicated below (note that you should also already have prepared the definition file). <JID> refers to the job number you were issued when you reserved the nodes.

```
> spattach –j <JID> –k cfxpdc.esy –i 2 double /home/definitionfiles/case1.def
```

Here we requested, in advance, **6** nodes for **120** minutes (`esubmit`). Once they had been allocated, we then logged on to one of the nodes (in this case node1.pdc.kth.se). We then chose to take advantage of **2** CPUs per node to **i**nteractively run the CFX **double** precision solver. Instructions will be given to the solver via the file /home/definitionfiles/**case1.def**

Voilá…it should be running. To confirm if CFX-Solver is actually running on the requested nodes, log on to one of them using `rxtelnet` and type `top`. Check the list to see if the CFX executable appears and the CPU usage. Type `q` to exit `top`.
Once the solver has finished, results will be saved to your working directory, unless otherwise specified.

## III.4 Running CFX on Dedicated Nodes – Batch Run (`esubmit`)

The difference between an `spattach` onto dedicated nodes and an `esubmit` is that the `esubmit` command is never reading any key stroke input. Instead it assumes all its input (if there is any) to come from a batch script. Once the batch script has been prepared and submitted, the job should run automatically after the nodes have been allocated.

Having already prepared your journal file, all you have to do is type:

```
> esubmit –n4 –t120 –c CAC cfxpdc.esy 2 double /home/definitionfiles/case1.def
```

Here we requested **4** nodes for **120** minutes (`esubmit`). We chose to take advantage of **2** CPUs per node to run the CFX **double** precision solver. Instructions will be given to the solver via the file /home/defintionfiles/**case1.def**

Again, to confirm if CFX is actually running on the requested nodes, log on to one of them using `rxtelnet` and type `top`. Check the list to see if the CFX executable appears and the CPU usage. Type `q` to exit `top`. A help menu has been included, so just type `cfxpdc.esy –h` to access it if you forget something.

## III.5 CFX Summary

The steps required to submit a CFX job on dedicated nodes with **interactive use** are as follows:

1) Generate a mesh (in ICEM, for example). Save as `something.msh`
2) Open CFX-Pre. Read in mesh and setup parameters (boundary conditions, models, discretization schemes, etc.). Create definition file `something.def` and FTP it to your working directory on PDC, along with the *.cfx and *.gtm files.
3) Login to whatever cluster you will be using on PDC. Confirm the lifetime of your tickets and increase it if necessary. Load necessary modules.
4) Copy the `cfxpdc.esy` script to your working directory. Make sure its permissions are set to executable.
5) Reserve nodes for interactive use in advance with the `esubmit` command. Once they have been allocated to you, logon to one of them with `rxtelnet` and perform the `spattach` of the `cfxpdc.esy` script.
6) Confirm if everything is running properly by logging on to one of the nodes and typing `top`.
7) Once your convergence criterion has been met an output file will be saved to your working directory, unless otherwise specified.

The steps required to submit a CFX **batch** job on dedicated nodes are as follows:

1) Generate a mesh (in ICEM, for example). Save as `something.msh`
2) Open CFX-Pre. Read in mesh and setup parameters (boundary conditions, models, discretization schemes, etc.). Create definition file `something.def` and FTP it to your working directory on PDC, along with the *.cfx and *.gtm files.
3) Login to whatever cluster you will be using on PDC. Confirm the lifetime of your tickets and increase it if necessary. Load necessary modules.
4) Copy the `cfxpdc.esy` script to your working directory. Make sure its permissions are set to executable and submit your job to PDC using the `esubmit` command.
5) Once the nodes requested have been allocated to you, confirm if everything is running properly by connecting to one of the nodes and typing `top`.
6) Once your convergence criterion has been met an output file will be saved to your working directory, unless otherwise specified.

# IV. Reminders

**Life time of tickets**

Always confirm the lifetime of your current tickets prior to submitting jobs:

Check the life time of your current tickets:

```
> klist –f
```

If tickets have short valid time, increase it:

```
> kinit –f –l <timetolive>
```

The *<timetolive>* must include both the queue waiting time and the execution time.

**Disk space**

Make sure you have enough disk space on PDC. Both CFX and Fluent files tend to be rather large, and results will not be written if enough disk space is not available. Refer to PDC online documentation for further information regarding disk storage options.

**Rules of thumb**

The following rules of thumb for parallel computations using both CFX and Fluent can be drawn:

- runs below 50 000 nodes should not be considered for parallel computation
- runs between 50 000 and 100 000 nodes can use up to 2 to 4 processors
- runs between 100 000 and 500 000 nodes can use up to 4 to 6 processors
- runs between 500 000 and 1 000 000 nodes can use up to 6 to 12 processors

**Environment variables (EASY)**

When writing your scripts, the following environment variables will come in handy:

| | |
|---|---|
| `SP_JID` | job ID |
| `SP_EASY_HOME` | home directory of EASY |
| `SP_SUBMIT_HOST` | node from which job was submitted |
| `SP_PROCS` | number of allocated nodes |
| `SP_NODES` | allocated nodes |
| `SP_HOSTFILE` | file that contains all allocated host names. There is one allocated host on each row |

# V. Notes. Suggestions

The PDC staff is extremely helpful and quick regarding problems that might appear, so contact them if things get ugly. I am extremely grateful to Ulf Andersson from PDC, for helping me with the initial scripting. Thank you.
I was (and still am) rather UNIX-ignorant but things have been improving. Nevertheless, I am sure you will be able to add new and improved features to the scripts presented here. Please communicate them so others can benefit. The PDC website http://www.pdc.kth.se is frequently updated (at least the "flash news" section) and contains loads of must-read information, so make sure to check it out before asking around.

CFD computations will surely be more and more common in the future at PDC. The KTH energy department, PDC (and others) will surely benefit from the following suggestions:

1 - perform benchmarking with Fluent and CFX (and other programs) on several clusters and report findings on website. This information will help future users to run there programs in a more efficient manner and at the same time provide PDC staff with valuable information regarding there cluster setups.

2 – Most computing centers provide some information on how to run the programs they have installed. The PDC website contains a section with the software available on PDC computers[1], and some basic information, but lacks a lot of important specifics. Fluent and CFX do not appear in this section.
Researchers who have no scripting knowledge whatsoever waste too much time figuring out things that a priori they should not need to worry about. Short seminars/courses on UNIX and shell scripting would benefit a lot of people. Setting up a **Forum** for PDC users on the PDC website also sounds like an interesting idea and would avoid always having to bother the PDC staff. Also, researchers should be encouraged to report there submission procedures for the different programs they run on PDC (as is done here) since most submission procedures are specific to the programs involved. A standard report format should be discussed and the reports published on the PDC website.

---

[1] http://schelly.pdc.kth.se/pdc/systems_support/software/