The User's Guide

INTRODUCTION TO NEC "TORI" MACHINE AT MÉTÉO-FRANCE

March 2007

(translated by J-A Maziejewski and endorsed by DSI))

(version 3,0)

CONTENTS

1.Presentation of the computer's architecture.

1.1.Introduction

The intensive computing system at Météo-France is composed of 2 NEC systems, each having 16 SX-8R vector nodes, a linux TX7 scalar front-end, a HP-UX batch handler and a GFS file server (also called "NAS head"). These two systems are fully identical. One of these system, called "sumo", is dedicated to operational tasks while the other one, called "tori" hosts R&D tasks. This latter one is described in this guide.

The configuration can be found on the web site of the IntraDSI under the heading:

DSI/SC -->DSI/SC/CC --> Configuration supercalculateur NEC pour Météo-France.

1.2.Acces point open to users

Users can work on the "tori" front-end and the 16 computing vector nodes.

Each of these vector nodes is a symmetrical multi-processor composed of 8 vector processors sharing a 128 Gb memory. The characteristics of a vector node are as follow:

- operating system: Unix system V (Super-UX)
- 8 processors at 35 Gflops theoretical maximum performance each. The R&D theoretical peak performance is therefore of 4,5 Tflops.

The nodes interconnect via a very efficient and high-performance internal crossbar network (IXS) with a maximum transfer rate of 2*8 Gb bidirectional/nodes.

The scalar front-end is the unique access entry point for users. The mainframe characteristics are as follow:

- Linux operating system (Suse)
- 16 cores Intel Itanium2 & 32 Gb memory.

Interactive work is only possible on the scalar front-end.

This front-end shall be used for compiling (cross compiler for SX8), for the submission of tasks on the vector nodes and for file transfer with either "cougar" or other access points of Météo-France's network. Tasks will preferably be launched in batch mode on the front-end (e.g. in the "compile" class). Only short tasks with low resources use will be tolerated in interactive mode (e.g. small editing jobs, library management, small compilation tests, file housekeeping).

Each computer is highly specialized. Computing batch tasks will be executed on the vector nodes, transfers via "ftserv" will be made from the front-end (cf II.4).

1.3.Disk configuration

The total user space disk is of 19 Tb. It is shared by all the vector nodes and the scalar front-end through the Global File System (GFS).

A local disk space of 256 Gb is available on each vector node, which can be used as a temporary working space for a "single-node" batch task (/localtmp). In this working space, data from a batch request will be automatically destroyed at the end of the request.

Anyone has access to this disk space, for node-local specific use.

HOMEDIR: user's permanent separate storage area. DSI/SC/CC provides the back up of data stored there. HOMEDIR user's data are saved by the software «Time Navigator». Users are split into 4 different file-systems (/cnrm, /mf, /ch, /ext) see 1.4 for allocation details. Total volume amounts to 6Tb.

TMPDIR: temporary disk space, accessible from every node, which a user can use either during the lifetime of his job or his interactive session. This space is therefore not saved, thus, all data are automatically destroyed at the end of the session (users themselves must transfer files to either their

workstations or to O3000/DMF). The explicit path of an TMPDIR is «/utmp». Its total volume amounts to 9Tb.

TMP_LOC: specific to a computing node: temporary disk space specific to a node. Be cautious, this space is unseen by other nodes and to the front-end. It may be of interest to use this space for a single-node job. The I/O efficiency there will be a lot better than on the shared TMPDIR.

FTDIR: buffer zone to be used when transferring through ftserv: see II.4 for advice about its use. This space is monitored by the system (automatic cleaning).

WORKDIR: intermediate (without back up) working space where data is stored as for long as possible. Most of the time, it is a buffer zone: there, users will frequently store retrieved data which is also archived elsewhere (generally on Origin3000/DMF). It is quicker to save files on WORKDIR than to go and retrieve them on «cougar». The oldest files are automatically destroyed as soon as the file system has reached a certain level. The explicit path of an WORKDIR is "/work". Total volume amounts to 4Tb.

Each of these disk space can respectively be reached using the variables $HOME, $TMPDIR or $tmpdir, $TMP_LOC or $tmp_loc, $WORKDIR or $workdir.

1.4. Allocation of file system on IMPE

The allocation of the different file systems GFS (visible from all nodes) is as follow:

TMPDIR FTDIR	/utmp	9,5Tb
WORKDIR	/work	4Tb
HOMEDIR	/cnrm /ch /mf /ext	2,5 Tb (CNRM) 2 Tb (operational suite) 1 Tb (Météo-France, non-operational) 1 Tb (non-MF users))

2. Services

2.1. Interactive

The interactive connexion is only allowed on the front-end ("telnet tori").

Only short tasks and resource use will be tolerated in interactive mode (short print and library works, very small compiling tests, file organization, batch job submissions). Big compiling test will have to be submitted in batch on this front-end (qsub -q compile).

2.2. Accounts

2.2.1. Login

An account name is made of 7 characters: *sgrpxxx*

- the letter s identifies the partner
  - m for Météo
  - c for Cerfacs
  - s for MERCATOR
  - e for outsiders
- grp is made of 3 letters used to identify projects or groups (a group in a unix sense is of the form *sgrp*).
- *xxx* is a string of 3 digits.

2.2.2.$HOME structure

the $HOME directory name has the structure /group/team/srpg/sgrpxxx with group being:

- cnrm for CNRM
- ch for operational suite
- mf for MF team not on the operational suite and not CNRM (DP, ENM, DIR,...)
- ext for non MF users (Cerfacs, Mercator, NEC, ...) and DSI

and "teams" one of:

- dp for Direction de la Production
- enm for ENM
- dir for all DIRs
- gc for GMGEC
- gc_ext for outside climate labs
- ge for GMME
- gi for GMEI
- gp for GMAP
- mr for MERCATOR
- cf for CERFACS
- dsi for DSI and NEC

Each user will have to transfer its own files from $HOME from "tora" (VPP) to the new NEC system. It was felt that it was not desirable to transfer all files from "tora" as most of them cannot be used on the new system (executables, reallocatable binaries, libraries,...). This will be the opportunity to do a big cleaning! The two systems will jointly operate for 6 months (till the end of June 2007), this period of time will give you the chance to transfer what you think is worth transferring.

2.2.3.Passwords

When creating an account, a temporary password is generated. It must be replaced when logging in for the first time. Then, it must be changed at regular interval within 12 weeks. A password is composed of at least 6 characters with no less than 1 special or numerical character and 2 alphabetic characters.

2.3.Batch

All tasks have to be launched in batch. The batch handler is NQSII.

2.3.1.Tasks submission

It is done directly from tori (the front-end).

A batch task (or batch request) can be single-processor, single-node (up to 8 processors) or multi nodes.

The task scheduler being based on real time (or elapsed time), it is absolutely necessary to specify this limit when setting the "qsub" submission options. **Please note that this is new, compared to the VPP procedure**.

For an optimal working of the scheduler, it is very important to describe as precisely as possible, tasks (number of nodes, number of processors per node, CPU time, elapsed time, node memory) and to make sure that options made through "qsub" are consistent with the mpirun command for jobs requiring MPI.

The main standard NQS instructions are as follow:

job submission:

qsub [options] myjob for submitting jobs (myjob = submission script)

ex:

qsub -q vector -b 2 -1 cputim_job=1200, cpunum_job=4, elapstim_req=600, memsz_job=12gb -j o . /test .sh

submits script "myjob" to the "vector" class, on 2 nodes, 4 procs per node, 12Gb memory per node, 1200 sec of CPU time per processor and 10 min total elapsed time.

"man qsub" gives more details about submission options.

Every submission option can be specified either on the « qsub » command line, or in the first lines of script « myjob », in which case they must be prefixed by « #PBS »:

```
#PBS -N JOBNAME                    # Name of the NQSII request
#PBS -q vector                     # NQS class
#PBS -T mpisx                      # type of job (if MPI -T mpisx)
#PBS -b 2                          # number of nodes used
#PBS -1 cpunum_job=2               # number of procs used/node
#PBS -1 cputim_job=00:16:00        # maximum CPU time
#PBS -1  memsz_job=12gb            # maximum memory size by node
#PBS -1 elapstim_req=00:10:00      # elapsed time (real time)
#PBS -j o                          # stdout and stderr on the same JOBNAME.nqsout file name
  ... job commands...
```

The four compulsory options to the scheduler are the following: elaptim_req, cpunum_job, memsz_job and -b. Should you not specify them, the default values will be used and therefore could not be suitable (weak values are set by default).

Job monitoring:

qstat [reqid] or qstat -f [reqid]

To monitor all jobs:

/usr/local/bin/qstat_all

To see the repartition of the reqid job on the different nodes:

qstat -J reqid

To stop/kill a job:

qdel reqid or qsig -9 reqid or qsig -SIGKILL reqid

To suspend/resume a request:

qsig -s STOP reqid and qsig -s CONT reqid

To hold/release a request:

qhold reqid and qrls reqid

Monitoring the output of running jobs:

qcat -o [reqid] for stdout or qcat -e [reqid] for stderr if different

2.3.2.Queue structure

This structure can be modified by the administrators in order to optimize the computer resources. The qstat -Q command gives all the defined queues. The actual queue structure is as follow:

NQSII queue	Executing machine	Number of nodes	Comments
compile	TX7 (tori)		Reserved for compilation jobs
ft	TX7 (tori)		Reserved for transferring files to/from Météo-France's network, especially « cougar ». CPU time is limited to 600 seconds.
vector	SX8	routing towards appropriate queue	Routes towards other queues (mono, express, multi) according to request in the number of procs and CPU time.
express	SX8	2, 3 or4 nodes max 8 procs/node	Multi-node & multiprocessors jobs with number of nodes ≤4 and elapsed time ≤ 3 hours Cannot be addressed directly. To use this queue, submit to « vector ».
mono	SX8	Only 1 node 8 procs maximum (1 proc by default)	Single node multiprocessors tasks Cannot be addressed directly. To use this queue, submit to « vector ».
multi	SX8	5,6,7 or 8 nodes max 8 processors by nodes	higher level multinodes multiprocessor tasks: 4 < number of nodes ≤ 8 Cannot be addressed directly. To use this queue, submit to « vector ».
debug	SX8	Max 2 nodes	Open on request for total view debugging

2.4. File transfer software to storage mainframe

The « ftserv » software has been implemented on the front-end only.

The local commands (ftpassword, ftget and ftput) have been implemented in order to regulate and to secure transfers between "cougar" and "tori". The transfers use the "ftp" protocole, login password on « cougar » is stored in an encrypted form in the config file « .ftuas » in the $HOME of tori.

« ftget » and « ftput » commands must be used from this machine to transfer files between "cougar" and $HOME or $WORKDIR or $FTDIR. The use of FTDIR is recommended should you not wish to keep the files after computing has been done.

These temporarily stored files will be seen from the vector nodes. $FTDIR, contrary to $TMPDIR, will be kept after the retrieval step, therefore stored files will be accessible by the request running on the vector nodes.

Usage:

Update the "ftuas" file using "ftmotpasse -u usercougar -h cougar-tori"

Ex: msys001@tori:/dsi/msys/msys001> ftmotpasse -u msys001 -h cougar-tori

Enter password for the machine: cougar-tori

New password:

Check:

2007/01/08 10:30:17 INFO pid(6371) update OK for the file '/dsi/msys001/.ftuas' for the remoteuser[msys001] on the remotehost[cougar-tori]

« Cougar-tori » is the name of the 10Gbit interface: the fast link between tori and cougar.

Use the "ftget" and the "ftput" commands to transfer from or to cougar. "ftput" command can be

used in an asynchronous mode (the job will resume without waiting for the end of the transfer). Enter "ftmotpasse", "ftget" or "ftput" for more details about the different options of these commands.

Ex: ftget –h cougar-tori fic_O3000 $FTDIR/fic_GFS

2007/01/08 10:00:31 INFO pid(8189) Debut de la demande: GET msys001@cougar-tori:fic_O3000 /utmp/ftdir/msys001/fic_GFS

2007/01/08 10:00:53 INFO pid(8189) Fin du transfert FT_OK: GET msys001@cougar-tori:fic_O3000 /utmp/ftdir/msys001/fic_GFS : Transfer complete.

During batch jobs, it is of the utmost importance that jobs/tasks should have the following structure (split into 3 NQS secondary jobs)

1. **Preprocessing step** (jobs running on the front-end "tori" in the "ft" class):
   i. file retrieval under GFS on $FTDIR buffer space (ftget)
   ii. qsub -q vector job_calcul
2. **Computing step** (job running on vector nodes):
   i. links of input files from $FTDIR to $TMPDIR
   ii. computation inside $TMPDIR
   iii. move output files (to be transferred to cougar) to $FTDIR
   iv. qsub -q ft post_processing
3. **Postprocessing step** (job running on the "tori" scalar front-end in the "ft" class):
   i. file archiving on cougar or any other local platform (ftput)

Exemple of a job sequence:

1. **Pre-fetch input files necessary for computation on FTDIR and launch the computation step**:

```
#JOB d'acquisition des fichiers
#PBS -N FILEGET
#PBS -j o
#PBS -o myoutput
#PBS -l cputim_l cputim_job=00:05:00
#PBS -l memsz_job=24mb
#PBS -q ft
set -x
cd $TMPDIR
ftget cougar-input $FTDIR/input
ls -l
cd $HOME/JOB
/usr/bin/nqsll/qsub $HOME/JOB/job.compute
```

2. **Computation job and launch of the archiving step:**

```
#PBS -N FILECOMPUTE
#PBS -j o
#PBS -b 1
#PBS -l cpunum_job=1
#PBS -l cputim_job=00:05:00
#PBS -l elapstim_req=00:05:00
#PBS -l memsz_job=240mb
#PBS -q vector
```

```
set -x
F_PROGINF=detail
export F_PROGINF
cd $TMPDIR
ls -l $FTDIR
ln -s $FTDIR/input fort.4
./a.out
mv fort.5 $FTDIR/output1
mv fort.6 $FTDIR/output2
ls -l
cd $HOME/JOB
/usr/bin/nqsll/qsub $HOME/JOB/job.put
```

3. **Save output on cougar**

```
#PBS -N FILEPUT
#PBS -j o
#PBS -l cputim_job=00:05:00
#PBS -l memsz_job=24mb
#PBS -q ft

set -x
cd $TMPDIR
ftput $FTDIR/output1 dircougar_experiment/output1
ftput $FTDIR/output2 dircougar_experiment/output2
echo "FIN"
```

2.5. HOMEDIR Back up

The software « Time Navigator » saves files which are on permanent space ($HOME). Lost files can be restored with the help of a user interface.

To restore a file, command « tina » must be launched on a client machine from the directory « hardy » (titan, forte, andante, adagio,...) and then, through the graphic interface, reach the NEC front-end using the following steps:

- To recover a file belonging to filesystem /ch or :ext:

  sauvegarde -> Systeme -> Connection -> tori

- To recover a file belonging to filesystem » /cnrm or /mf:

  sauvegarde -> Application -> Connection -> tori.fs

  Then a user interface appears asking for your login and password for tori. You will then see your home directory on tori, you can unfold the directory tree and from the left side window, you can go back. Choose the files to be recovered (tick next to the filename) then:

  Sauvegarde -> Restaurer

2.6. Application environment

2.6.1. The FORTRAN Compiler

FORTRAN 95 is the norm for the FORTRAN compiler (sxf90). It is a "cross compiler" which generates codes for vector nodes from the front-end "tori".

« sxf90 » enables to get to the cross compiler, either run « man sxf90 » or read the guide to know the different options (see II.7.2) or the full documentation « cookbook ».

The optimization options are the following:

vector and scalar optimisation :

- `SXf90 -C` **`hopt`** `code.f` **highest level** (quite risky)
- `SXf90 -C` **`vopt`** `code.f` **high level** (why not!)
- `SXf90 -C` **`vsafe`** `code.f` **safe vectorization** (if you say so...)

scalar only optimisation:

- `SXf90 -C` **`sopt`** `code.f`
- `SXf90 -C` **`ssafe`** `code.f`
- `SXf90 -C` **`debug`** `code.f` (last resort...)

Debugging:

- -C debug suppresses every optimization and every vectorization.
- -gv produces a table of symbols for debugging with vectorization info
- -eC check for out of bounds array indices
- -eR check the array syntax

Type représentation:

SX8R can compute in 32 bits or 64 bits.

- `SXf90` **`-dw`** `code.f90` computations on 32 bits
- `SXf90` **`-ew`** `code.f90` or on 64 bits

	-dw	-ew
INTEGER*2	integer type(2bytes) <*>	integer type(8 bytes)
INTEGER[*4]	integer type(4 bytes)	integer type(8 bytes)
INTEGER(KIND=2)	integer type(2 bytes)<*>	integer type(8 bytes)
INTEGER(KIND=4)	integer type(4 bytes)	integer type(8 bytes)
LOGICAL*1	logical type(1 bytes)<*>	logical type(8 bytes)
LOGICAL[*4]	logical type(4 bytes)	logical type(8 bytes)
LOGICAL(KIND=1)	logical type(1 bytes)<*>	logical type(8 bytes)
LOGICAL(KIND=4)	logical type(4 bytes)	logical type(8 bytes)
REAL[*4]	real type(4 bytes)	real type(8 bytes)
REAL*8	real type(8 bytes)	real type(8 bytes)

	-dw	-ew
REAL*16	real type(16 bytes)	real type(16 bytes)
REAL(KIND=4)	real type(4 bytes)	real type(8 bytes)
REAL(KIND=8)	real type(8 bytes)	real type(8 bytes)
REAL(KIND=16)	real type(16 bytes)	real type(16 bytes)
DOUBLE PRECISION	real type(8 bytes)	real type(8 bytes)
COMPLEX[*8]	a pair of real type(4 bytes)	a pair of real type(8 bytes)
COMPLEX*16	a pair of real type(8 bytes)	a pair of real type(8 bytes)
COMPLEX*32	a pair of real type(16 bytes)	a pair of real type(16 bytes)
COMPLEX(KIND=4)	a pair of real type(4 bytes)	a pair of real type(8 bytes)
COMPLEX(KIND=8)	a pair of real type(8 bytes)	a pair of real type(8 bytes)
COMPLEX(KIND=16)	a pair of real type(16 bytes)	a pair of real type(16 bytes)
1.23E1	real type(4 bytes)	real type(8 bytes)
1.23D1	real type(8 bytes)	real type(8 bytes)
1.23Q1	real type(16 bytes)	real type(16 bytes)
1.23_4	real type(4 bytes)	real type(8 bytes)
1.23_8	real type(8 bytes)	real type(8 bytes)
1.23_16	real type(16 bytes)	real type(16 bytes)
SQRT	real type(4 bytes)	real type(8 bytes)
DSQRT	real type(8 bytes)	real type(8 bytes)
QSQRT	real type(16 bytes)	real type(16 bytes)

<*> the size is 4 bytes when **-eW** is present. If one wants to keep storage on 2 and 1 bytes respectively, to integer (kind=2) and logical (kind=1), option **-d W** must be specified.

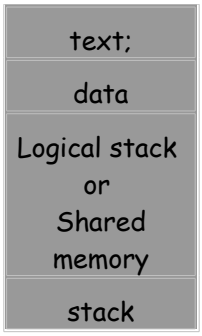
If with -dw one wants to transform REAL in REAL*8, it is necessary to add:

```
SXf90 -dw -Wf"-A dbl4" code.f
```

The association of these different options of compilation provides a very complete representation of numbers.

Dynamic and statistic allocation

Memory map:



- stack: limited by default to 512 Goctets (or to the maximum of the memory)
- data: static memory zone (keeping local values <-> loss of space memory)

stack and heap can be either set to zero or undefined (NaN, which is useful for debugging) :

SXf90 -P stack **-Wf"-init stack=zero heap=zero"** code.f

Furthermore, common and modules variables (including derived types) can also be initialized at the beginning of the job. Must be specified when editing links:

SXf90 -Wl"-f zero" -o code.x code.o

By default, the static mode is applied and variables are initialized at zero. Almost as in the VPP. It is essential to thoroughly check the code, specially when parallel programs, by ensuring the variables are correctly initialized. This can be done in four steps.

```
SXf90 -P static -o code.x code.f90
SXf90 -P stack -Wf"-init stack=zero heap=zero" -Wl"-f zero" -o code.x code.f90
SXf90 -P stack -Wf"-init stack=zero heap=zero" -Wl"-f nan" -o code.x code.f90
SXf90 -P stack -Wf"-init stack=nan heap=nan" -Wl"-f nan" -o code.x code.f90
```

The cost of these initializations can be high. Therefore, once the code has been corrected, one must make sure that the software can run without it.

Compilation options:

syntaxe f90 NEC	Description
f90	compiler
-p	activate profiling mode. place it simply to the link edition
-D*define*	*define* statement for cpp
-U*define*	cancel *define* for cpp
-F	generate the file i.*fichier*.f resulting from pre compilation cpp of *fichier*.f
-P static	local variables created in a static way
-P stack	local variables created in the "stack" space
-P auto	auto tasking
-P multi	parallelization using directives
-Wf'-O nooverlap'	show the compiler that dynamic table (POINTER) do not overlap => vectorisation
-Wf'-pvctl noassume'	interpret a declared "dummy argument" TAB(1) as a vector
-Wf'-pvctl loopcnt=1000000'	set the maximum size of temporary tables

syntaxe f90 NEC	Description
f90	compiler
-Wf'-pvctl vwork=stack'	allocate temporary vectors in the "stack" space
-Wf'-ptr word'	the word is the unit for the dynamic allocation of memory
-Wf'-ptr byte'	the octet is the unit for the dynamic allocation of memory
-Wf'-O extendreorder'	optimise the scalar code
-Wf'-init stack=[zero|nan]	initialize the "stack" space to 0 or Not A Number
-Wf'-init heap=[zero|nan]	initialize "heap" space to 0 or Not A Number
-Wf'-Z *n*'	expand the dynamic zone to *n* octets
-Wf'-prob_generate'	create a profile when executing the code
-Wf'-prob_use'	use the previous profile to optimize the code

2.6.2. Main differences with the VPP

- User's work (edit, compiling, linking, library creation…) should be done on the front-end (TX7). The commands to be used are sxf90 (compilation), sxld (link edition) et sxar (SX8 format binary libraries). Then, executables are run on one or several nodes.
- Contrary to the VPP, a MPI parallel executable must be launched with the « mpirun » command.
- Contrary to the VPP, by default, variables are not initialized to zero.
- Equivalence of compilations options between the several systems (source E. Gondet /MERCATOR)

Compaq	IBM SP3	Fujitsu VPP5000	Nec SX	PGI	Sgi O2K	Description
F90	xlf[90][_r]	frt	f90	pgf90	f90	Compiler's name
-c						Sole Compilation
-o *executable*						Creation of an executable
-I *directory*						Directory where to seek "INCLUDE"
-L *directory*						Directory where to seek libraries
-l *biblio*						link lib*biblio*.a
-version|-what	-#	-v	-v	-V or -flags	-version	Number of the compiler version
-automatic (omp défaut)	-qnosave	-K auto	-Pstack (défaut)	-Mnosave	(défaut)	Mode "stack"= allocation of the local variables in the stack
-noautomatic (défaut)	-qsave	(defaut)	-Pstatic	-Msave	-static	Static mode= allocation of the local variables in the bss
-old_f77|-f77	-qfixed=72 -qsuffix=f=f	-Fixed -X7 (-X7 -> strict F77 norm)	-f0	-Mnofree	-fixedform	Set format, usually by default for file in .f

Compaq	IBM SP3	Fujitsu VPP5000	Nec SX	PGI	Sgi O2K	Description
F90	xlf[90][_r]	frt	f90	pgf90	f90	Compiler's name
(default) | -free toto.f	-qfree	-Free -X9	-f4	-Mfree	-freeform (default pour fichier en toto.f90)	Free format and with F90 norm usually by default for file in .f90
-fixed toto.f90	-q fixed=72 -q suffix=f=f90 toto.f90	-Fixed -X9	-f3	-Mnofree toto.f90	-fixedform toto.f90	Set format **but** with F90 norm
(default)	-qmoddir=. (défaut)	-Am	(default)	(default)	(default)	Compilation with F90 modules in the local directory
-qmodule *directory*	-qmoddir = *directory*	-Am – I *directory*	-I *directory*	-module *directory*	-I *directory*	Compilation with F90 modules in another directory

2.6.3. Available libraries

The mathematical libraries optimized by NEC for SX8 are available on tori: /usr/local/SX/lib/MathKeisan/lib. Especially libblas.a, libfft.a, liblapack.a, libparblas.a, libparfft.a, libsblas.a. They must be explicitly specified during the linking.

Local libraries (GRIB, BUFR, NETCDF…) are reached by the command /usr/local/SX/lib.

2.6.4. Environnement variables

$F_PROGINF = [no|yes|detail]	Activates CPU profiling
$F_FILEINF = [no|yes|detail]	Activates I/O profiling
$F_RECLUNIT = [word|byte]	Select direct access for unit RECL
$F_NORCW	Suppresses control words in sequential binary files
$F_SETBUF	Sets I/O buffer size

2.7. Miscellaneous

2.7.1. Accounting utilities

To get an idea about the use of the computer resources, set to « detail » the following variable: F_PROGINF , F_FILEINF and MPIPROGINF

WARNING: Be careful, these variables will only evaluate the resources used by a Fortran code but will not evaluate the total resources used by the batch request.

The command « /usr/local/bin/ja » executed at the end of a script, gives an overall picture of the resources used by the request.

It is expected that a more appropriate new command, typically of the form "ja" will be available soon.

2.7.2. Documentation

Two NEC guides can be found on line on intranet under http://intradsi.meteo.fr/:

DSI/SC-->DSI/SC/CC-->Guides utilisateurs de DSI/SC/CC-->documentation NEC

DSI/SC-->DSI/SC/CC--> Guides utilisateurs de DSI/SC/CC-->cookbook