LHC COMPUTING GRID

MAUI COOKBOOK FOR LCG

Document identifier:	
EDMS id:	
Version:	**v1.1**
Date:	
Section:	**IT-GD-GIS**
Document status:	**Draft**
Author(s):	Sophie Lemaitre, Jeff Templon, Steve Traylen, Markus Schulz, Davide Salomoni
File:	**Maui-Cookbook**

Abstract: *This document introduces the Maui advanced job scheduler in the context of LCG. It also shows how Maui is being used at two sites having different approaches: the English Rutherford Appleton Laboratory (RAL) and the Dutch National Institute for Nuclear and High Energy Physics (NIKHEF).*

CONTENTS

1. OVERVIEW

Maui is an advanced job scheduler. It is an optimized and configurable tool capable of supporting a large array of scheduling policies, dynamic priorities, extensive reservations, and fairshare.[1]

The official Maui web site is :
http://www.supercluster.org/maui

As said there, Maui is a "community project and may be downloaded, modified, and distributed. It has been made possible by the support of Cluster Resources, Inc and the contributions of many individuals and sites including the U.S. Department of Energy, the Pacific Northwest National Laboratory (PNNL), the Center for High Performance Computing at the University of Utah (CHPC), San Diego Supercomputing Center (SDSC), MHPCC, Brigham Young University (BYU), and NCSA."

This document aims at providing a brief overview of the maui capabilities, and describes some common use cases that can be frequently be encountered in (among others) grid environments. More detailed documentation can be found on the Maui web site, **http://www.supercluster.org/maui**; the "Maui Administrator's Guide" (**http://www.clusterresources.com/products/maui/docs/mauiadmin.pdf**) is especially helpful.

[1] Fairshare is a mechanism which allows historical resource utilization information to be incorporated into job feasibility and priority decisions.

2. MAUI IN THE LCG CONTEXT

Maui can be useful in the LCG context for several reasons:

- It can be used as the job scheduler for several batch systems currently supported in LCG: OpenPBS, PBSPro, Torque, Condor, etc.
- It can control how many resources you want to dedicate to each Virtual Organisation (VO) allowed to run jobs at your site. For instance, you can define that the ”cms” VO cannot use more than 60% of the total CPU, whereas the ”atlas” VO can only use up to 30% (with the remaining resources being shared perhaps among other VOs).
- It allows to reserve resources for a given user or group (and thus VO).
- It allows to limit the maximum resources that can be used by a given user or group.
- It supports (possibly complex) fairshare policies.

Warning : The two batch systems Torque and OpenPBS are incompatible. In other words, the whole farm of a site should be running either Torque or OpenPBS.

3. INSTALLING MAUI

3.1. RPMS

To install Maui, the following RPMs are needed :

- maui
- maui-client

The Maui RPMs are available for RedHat 7.3 and for Scientific Linux 3 (SL3) or SL3 binary-compatible distributions (like CentOS).

For LCG, they can be downloaded from the following web site (provided by Steve Traylen) :
http://hepunx.rl.ac.uk/~traylens/rpms/maui/

Install the RPMs as root :

```
> rpm -Uvh maui*.rpm
```

3.2. STARTING MAUI

To start Maui, use the following command :

```
> service maui start
```

N.B.: the ”restart” command stops Maui, but doesn’t actually restart Maui. Use start/stop instead.

3.3. SETTING MAUI AS THE JOB SCHEDULER

By default, batch systems like OpenPBS or Torque have their own job scheduler. Normally, this is much more simple-minded than Maui; for example, in the case of OpenPBS and Torque, the default is a simple `fifo` scheduler. Therefore, before Maui can be used, the batch system must be told to use it.

With OpenPBS or Torque, this is accomplished setting the `scheduling` state of the server to `False`:

```
[root@tbn09 root]# qmgr -c "set server scheduling = False"
[root@tbn09 root]# qmgr -c "list server"
Server tbn09.nikhef.nl
        server_state = Idle
        scheduling = False
[...]
```

4. MAUI'S CONFIGURATION FILE

The scheduling policies are defined in Maui's configuration file :

```
/var/spool/maui/maui.cfg
```

Here is the default configuration file:

```
#
# MAUI configuration example
# @(#)maui.cfg David Groep 20031015.1
# for MAUI version 3.2.5
#
SERVERHOST              localhost
ADMIN1                       root
ADMINHOST               localhost
RMTYPE[0]                  PBS
RMHOST[0]                  localhost
RMSERVER[0]              localhost

SERVERPORT            40559
SERVERMODE            NORMAL

# Set PBS server polling interval. Since we have many short jobs
# and want fast turn-around, set this to 10 seconds (default: 2 minutes)
RMPOLLINTERVAL        00:00:10

# a max. 10 MByte log file in a logical location
LOGFILE               /var/log/maui.log
LOGFILEMAXSIZE        10000000
LOGLEVEL              3
```

You might then want to modify it according to your site policy (see the "Configuration Examples" section for some common use cases).

N.B.: every time you modify the configuration file you need to restart Maui for the configuration to take effect. See the section on Admin Commands for how to do it.

5. USEFUL ADMIN COMMANDS

Here are some useful commands, that must be run as root :

- To start/stop Maui :

```
> service maui start
> service maui stop
```

  N.B.: the ”restart” command stops Maui, but doesn’t actually restart Maui. Use start/stop instead.

  N.B.: since Maui acts as the job scheduler for the underlying batch system, there is obviously interaction between the batch system and Maui. In practice, if for whatever reason you need to restart the batch system (e.g. Torque or OpenPBS), it is advisable to restart Maui as well. A recipe for the restart of the batch system is thus the following:

  – stop maui
  – restart the batch system (OpenPBS or Torque)
  – start maui

- To see what the scheduler is actually doing and check current reservations :

```
> showres -n
```

- To see the fairshare usage rates of different users/groups :

```
> diagnose -f
```

  Sample output:

```
[root@tbn18 root]# diagnose -f
FairShare Information

Depth: 24 intervals   Interval Length: 1:06:00:00   Decay Rate: 0.99

FS Policy: DEDICATEDPES
System FS Settings:  Target Usage: 0.00    Flags: 0
[...]
GROUP
-------------
atlas*           37.87  50.00    13.68    6.32   12.67   22.97   76.85   96.41   98.94   9
6   99.51   96.52   97.53   77.58   10.05    6.78    2.85    3.79    5.30    4.22    7.26
[...]
```

The command `diagnose -f` outputs a lot of information, the interpretation of which depends on how fairshare was configured. In the excerpt above we see that users belonging to the group `atlas` currently have a fairshare value of 37.87, out of a fairshare target of 50. This fairshare value was built taking into account a number of historical information in the past. Simplifying, this means that users belonging to the group `atlas` will be allocated resources in the coming iterations of the scheduler, while users belonging to groups having already reached their fairshare target will be pushed back.

- To see in which order the jobs are going to be exectuted :

```
> showq
```

Sample output:

```
[root@tbn18 root]# showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME

179857             dteam001    Running     1     4:59:59  Fri Jan 28 15:50:29
178919             atlas021    Running     1    10:26:22  Wed Jan 26 02:16:51
179078             atlsm003    Running     1  1:02:46:59  Wed Jan 26 18:37:28
179103             atlsm003    Running     1  1:03:07:31  Wed Jan 26 18:58:00
[...]

   242 Active Jobs     240 of  240 Processors Active (100.00%)
                       120 of  120 Nodes Active      (100.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

179795              lhcb002       Idle     1  3:00:00:00  Fri Jan 28 14:04:27
179797              lhcb002       Idle     1  3:00:00:00  Fri Jan 28 14:27:18
179799             atlas034       Idle     1  3:00:00:00  Fri Jan 28 14:30:34
[...]
47 Idle Jobs

BLOCKED JOBS----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

177077             pvier000       Idle    15     4:59:59  Fri Jan 14 16:27:31

Total Jobs: 290   Active Jobs: 242   Idle Jobs: 47   Blocked Jobs: 1
```

In this example one can see that there are several jobs running (actually filling up all the available CPUs), several others in `idle` mode (which means that they will be running as soon as resources

become available), and one job in `blocked` mode (which means that the job has been blocked by some policy).

- To show a concise summary of the system state :

```
> showstats
```

Sample output:

```
[root@tbn18 root]# showstats

maui active for   17:06:55:18  stats initialized on Thu Jan  1 01:00:00

Eligible/Idle Jobs:                   52/104       (50.000%)
Active Jobs:                         242
Successful/Completed Jobs:        132463/298673    (44.351%)
Avg/Max QTime (Hours):              3.00/231.05
Avg/Max XFactor:                    0.02/7.77

Dedicated/Total ProcHours:     1101354.40/1499639.78 (73.441%)

Current Active/Total Procs:          240/240       (100.000%)

Avg WallClock Accuracy:           14.127%
Avg Job Proc Efficiency:          92.364%
Est/Avg Backlog (Hours):            -2.89/0.00
```

- To show a statistical listing of nodes and memory :

```
> showstats -n -S
```

Sample output:

```
[root@tbn18 root]# showstats -n -S


Memory Requirement Breakdown:

  Memory  Proc Percent  InitialNH Percent  ProcHours Percent
       1     1    0.75      20950 13400.01        156  100.00
     501    47   35.07          0    0.00       7348  100.00
    1006    60   44.78          0    0.00       9381  100.00
    2016    26   19.40          0    0.00       4065  100.00
   TOTAL   134  100.00      20950  100.00      20950  100.00
```

```
Node Statistics

Summary:    1    1MB Nodes   99.82% Avail   89.56% Busy  (Current:   0.00% Avail    0.00%
Summary:   47  501MB Nodes   99.23% Avail   80.66% Busy  (Current:  97.87% Avail   97.87%
Summary:   60 1006MB Nodes   97.21% Avail   77.33% Busy  (Current:  80.00% Avail   78.33%
Summary:   26 2016MB Nodes   99.69% Avail   83.33% Busy  (Current: 100.00% Avail   96.15%
System Summary:  134 Nodes   98.40% Avail   79.74% Busy  (Current:  89.55% Avail   88.06%
```

Leaving out the `-S` flag shows details for individual nodes, as in

```
node1-103.farmnet.nikhef.nl    99.85%    86.00%      Busy
```

- Other useful `showstat` commands :

```
> showstats -g
> showstats -u
```

These commands shows a statistical listing of all active groups and users, respectively.

6. CONFIGURATION EXAMPLES

By way of example, here are the Maui configuration files implemented at the following sites : - the Rutherford Appleton Laboratory (RAL) located in the UK, - the National Institute for Nuclear Physics and High Energy Physics (NIKHEF) located in the Netherlands.

6.1. MAUI AT RAL

Below is the current Maui configuration at RAL.
It is commented in details.

This configuration :

- defines different fairshare targets for all the supported VOs,
- reserves two nodes for the short queue.

```
#
# MAUI configuration example
# for MAUI version 3.2.5
#

SERVERHOST lcgce02.gridpp.rl.ac.uk
ADMIN1 root
ADMINHOST lcgce02.gridpp.rl.ac.uk
RMCFG[base]             TYPE=PBS

SERVERPORT              40559
SERVERMODE              NORMAL

# Set PBS server polling interval. Since we have many short jobs
# and want fast turn-around, set this to 1 minute (default: 2 minutes)
RMPOLLINTERVAL          00:01:00

# a max. 5 MByte log file in a sensible location
LOGFILE                 /var/log/maui.log
LOGFILEMAXSIZE          5000000
LOGLEVEL                1
LOGFILEROLLDEPTH        5

############################################################
# Set up fairshares
# To diagnose fairshares on a running system use `diagnose -f`.
```

```
# We will consider the last 7 24 hour periods for our fair
# share calculations. The influence of each 24 hour period
# decreases by a factor of 0.8 each time.
FSPOLICY                DEDICATEDPS
FSDEPTH                 7
FSINTERVAL              24:00:00
FSDECAY                 0.8
FSWEIGHT                1
FSUSERWEIGHT            5
FSGROUPWEIGHT           30

# Suppose we have 1000 cpus that can run jobs.
# We want to devide this up as percentages allocated to
# each VO or unix group.

# We specify a FSTARGET of this %. It will be easier in the first case
# if these numbers add to 100% but this is not required.

# For each group we also specify both a soft and hard limit (soft,hard) for
# the number of CPUs a group may use. A group can never use more than its hard limit.
# In this case we have permitted the whole farm to be filled by one group.
# The soft limit is only enforced as long as another group
# is not allready at their soft limit. This results in a group currently below their
# allocation will reach it with complete priority over groups already above their soft
# limit.

GROUPCFG[dteam]  FSTARGET=1     MAXPROC=10,1000
GROUPCFG[alice]  FSTARGET=50    MAXPROC=500,1000
GROUPCFG[atlas]  FSTARGET=20    MAXPROC=200,1000
GROUPCFG[cms]    FSTARGET=1     MAXPROC=10,1000
GROUPCFG[babar]  FSTARGET=1     MAXPROC=10,1000
GROUPCFG[lhcb]   FSTARGET=1     MAXPROC=10,1000
GROUPCFG[dzero]  FSTARGET=10    MAXPROC=100,1000
GROUPCFG[hone]   FSTARGET=11    MAXPROC=110,1000
GROUPCFG[zeus]   FSTARGET=5     MAXPROC=50,1000

# Try and do some fairshare between all users.
# This is bit hard because pool accounts can
# change ownership every 10 days. This is why we set FSUSERWEIGHT
# a lot lower than FSGROUPWEIGHT above to make this less significant
# than the fairshare amongst groups.
USERCFG[DEFAULT]    FSTARGET=10+

# Give some extra priority to short jobs over long jobs.
```

```
# Read the manual to try and understand this. Good Luck.
XFACTORWEIGHT    1

# By setting this to  zero we instruct the scheduler to not behave in
# a FIFO way at all.
QUEUETIMEWEIGHT   0

# Make sure we don't allow queue stuffing though when people exceed
# the MAXPROC values.
JOBPRIOACCRUALPOLICY  FULLPOLICY

# Make sure that jobs with a negative priority are properly handled
ENABLENEGJOBPRIORITY true
REJECTNEGPRIOJOBS false

# Favor the fastest available CPU for incoming jobs.
NODEALLOCATIONPOLICY PRIORITY
NODECFG[DEFAULT]  PRIORITYF='SPEED'

# Now and again Maui defers a job from running that it
# could not start for some reason, I don't understand why?
# Setting this to 0 stops the job being defered.
DEFERTIME        0

########################################################################
#  Create a reservation of two (TASKCOUNT) nodes for the short queue.
#  Use the commands `showres -n` and `diagnose -f` to diagnose this.
SRCFG[twonode4S]        STARTTIME=0:00:00 ENDTIME=24:00:00
SRCFG[twonode4S]        PERIOD=INFINITY
SRCFG[twonode4S]        TASKCOUNT=2 RESOURCES=PROCS:1;MEM:400
SRCFG[twonode4S]        CLASSLIST=short
```

6.2. MAUI AT NIKHEF

Below is the Maui configuration at NIKHEF.

NIKHEF has to deal with many different VOs and with local users as well. The farm at NIKHEF is very heterogeneous (machines with different clock speeds), and this has to be takein into account in the scheduler configuration.

Part of the NIKHEF farm is funded by the Dutch NCF project. Therefore, as a policy, NCF should always have as many resources available for their own jobs as the equivalent number of nodes they contribute to the farm.

```
#
# MAUI configuration for the NIKHEF NDPF
# Cluster: lcgprod
#

# Host name of the machine where Maui runs (this parameter MUST be specified !)
SERVERHOST tbn18.nikhef.nl

# Maui has 3 levels of admin access :
# - ADMIN1 : full control of all Maui functions (the first user in the list is the primary
#            admin, and Maui is running under its ID).
# - ADMIN2 : allowed to change all job attributes and granted access to all informational
#            Maui commands.
# - ADMIN3 : allowed access to all informational Maui commands. Cannot change scheduler or
#            job attributes
ADMIN1 root davidg davides templon
ADMIN3 fokke

# List of hosts from which any Maui administrative command can be run
ADMINHOST tbn18.nikhef.nl tbn04.nikhef.nl localhost.localdomain localhost

RMTYPE[0]            PBS
RMHOST[0]      tbn18.nikhef.nl
SERVERPORT           40559

# There are 3 possible running modes for Maui :
# - SIMULATION : the scheduler does not background itself as in TEST and NORMAL mode.
# - TEST : to test Maui (the jobs are not really scheduled nor submitted, but Maui still
#          collects real time job and node information as if they were scheduled)
# - NORMAL : all the Maui functionalities are enabled
SERVERMODE             NORMAL

# PBS server polling interval : how often Maui will refresh its Resource Manager information
```

```
RMPOLLINTERVAL          00:01:30

# Log file of maximum 50 MB
LOGFILE                 /var/log/maui.log
LOGFILEMAXSIZE          50000000

# Amount of information is actually logged (from 0 to 9, 9 being the most verbose)
LOGLEVEL                4

# Number of old logs maintained (1 by default)
LOGFILEROLLDEPTH        5

NODESETPOLICY ONEOF
NODESETATTRIBUTE FEATURE
# The following parameter allows us to define node sets.
# The string following NODESETLIST specifies the different
# subclusters of the NIKHEF farm; the nodes belonging to
# each subclusters have the approriate tag (in this case,
# "dzero", "halloween", or "ncf") defined in the Torque
# node database.
NODESETLIST dzero halloween ncf
NODESETDELAY 0:00:00
NODESYNCTIME            0:00:05

# All the nodes are available for all the users/groups.
# Note that if you specified SINGLEUSER this would mean
# that more than one job could run on a given node
# if and only if owned by the same user, which is something
# we don't want.
NODEACCESSPOLICY      SHARED

NODEAVAILABILITYPOLICY  DEDICATED:PROCS
NODELOADPOLICY          ADJUSTPROCS
DEFERTIME               0
JOBMAXOVERRUN           0

# The jobs with a negative priority are not rejected
REJECTNEGPRIOJOBS       FALSE

# This parameter is very important in the case of a heterogeneous cluster.
# It specifies the header used to extract node processor speed via
# node attributes. We mark all nodes in the Torque database with the
# tag "xpsXXXX", where XXXX is the cpu speed. For example, P-III 800MHz
# nodes are marked "xps800", while P-IV 2.8GHz nodes are marked
```

```
# "xps2400". This allows maui to take the CPU speed value into account
# when making policy decisions.
FEATUREPROCSPEEDHEADER  xps

# Policies
BACKFILLPOLICY          ON
BACKFILLTYPE            FIRSTFIT

# This specifies how Maui will allocate resources
NODEALLOCATIONPOLICY    FASTEST

RESERVATIONPOLICY       CURRENTHIGHEST
RESERVATIONDEPTH        12

##################
# Global Weights

# Job's queuetime priority factor (1 by default). Here, a job that has been
# queued for 120 minutes will have a queuetime priority factor of 2*120
QUEUETIMEWEIGHT 2

XFWEIGHT 10
XFCAP                   100000
RESWEIGHT 10

CREDWEIGHT       9
USERWEIGHT       10
GROUPWEIGHT      10

FSWEIGHT         90
FSUSERWEIGHT 1
FSGROUPWEIGHT 10
FSACCOUNTWEIGHT         10

########################
# FairShare parameters #
########################

# use dedicated CPU ("wallclocktime used") metering
# Decays by a factor of 0.99 over FSDEPTH*FSINTERVAL = 24*30 hours = 30 days
FSDEPTH                 24
FSINTERVAL              30:00:00
FSDECAY                 0.99
```

```
# Unit of tracking fairshare usage :
# - DEDICATEDPS tracks dedicated processor seconds
# - DEDICATEDPES tracks dedicated process-equivalent seconds
FSPOLICY              DEDICATEDPES

# Maximum value for a job's total pre-weighted fairshare component
FSCAP                 100000

#########################################################################
# Fair shares and limits
#
# Policies to implement
# * ANTARES should get 5THzEquivHours/month (as per directive KarelG,20040803)
# * THEORY should get a 10 THzEquivHours/month (as per directive KarelG,20040908)
#   -- that would mean 7 2GHz CPUs continuously for 1 month
# * LHC should get "the rest"
# * We must honour NCF/NL-Grid jobs on at least 120 CPUs (size of the NCF farm)
# * 10% of the resources is for test/dteam/health-monitoring
#
# Relationship of priorities:
#  CREDWEIGHT 9% (0.5 u/0.5 g); FSWEIGHT 98% (0.5 u/0.5 g); QTIME 1%
#
# To get reasonably fair scheduling, there should be a free slot every, say,
# 15 minutes. That means that, with 250 CPUs, the maxwalltime should be
# 15*250 min = 62 hrs, so "qlong" is already somewhat long.
# But, since we will allocate at random slow and fast nodes (slowest=0.8GHz
# and we have on average 340 GHzEquiv now, so avg speedratio=1.7), we can
# tolerate a queue size of up to 62*1.7 hrs = 105 hrs.
#
# Thus, the "qlong" queue of 96 hours is fine, but the infinite queue must
# be phased out again (it was abused anyway and never fulfilled the
# original purpose).
#
# The ratio between NLGrid(NCF) usage and internal usage should be
# according to the allocations made by the NCF Steering Group. Today,
# 48% of our resources is NCF funded (in GHz), and thus the nlgrid Account
# should have a 48% fair-share. But again, LCG is part of NL-Grid as well,
# so we should account for that somehow.

USERCFG[DEFAULT]      MAXJOBQUEUED=350
GROUPCFG[DEFAULT]     FSTARGET=0      PRIORITY=1        MAXPROC=10

# There are three Fair Share classes : "lhc" (with 50% fairshare), "nlgrid"
# (with 50% Fairshare) and "niklocal" (with the default 0% Fairshare)
```

```
#
# Note that we aggregate several groups/users under a common account
# header (e.g. alice,atlas,etc are all parts of the "lhc" group, see
# parameter ADEF below).

ACCOUNTCFG[lhc]      FSTARGET=50                      MAXPROC=279
ACCOUNTCFG[niklocal]                                  MAXPROC=250
ACCOUNTCFG[nlgrid]   FSTARGET=50                      MAXPROC=120

CLASSCFG[qinfinite]                 PRIORITY=1

# The limits applied appear to be a MIN() of all applicable limits, so e.g.
# since alice001 is not mentioned by name, its FSTARGET is MIN(1,40) = 1
# where the "1" is from DEFAULT USERCFG and the "40" is from alice GROUPCFG.

GROUPCFG[users]      FSTARGET=0      PRIORITY=10       MAXPROC=50
GROUPCFG[tmpusr]     FSTARGET=0-     PRIORITY=10       MAXPROC=2
GROUPCFG[tbadmin]    FSTARGET=10     PRIORITY=5000     MAXPROC=200
GROUPCFG[dteam]      FSTARGET=2      PRIORITY=5000     MAXPROC=32
GROUPCFG[tutor]                      PRIORITY=1000000  MAXPROC=42

GROUPCFG[alice]      FSTARGET=15     PRIORITY=100      MAXPROC=240     ADEF=lhc
GROUPCFG[atlas]      FSTARGET=50     PRIORITY=100      MAXPROC=250     ADEF=lhc
GROUPCFG[atlsgm]     FSTARGET=50     PRIORITY=100      MAXPROC=250     ADEF=lhc
GROUPCFG[lhcb]       FSTARGET=35     PRIORITY=100      MAXPROC=250     ADEF=lhc
GROUPCFG[lhcbsgm]    FSTARGET=35     PRIORITY=100      MAXPROC=250     ADEF=lhc
GROUPCFG[cms]        FSTARGET=1-     PRIORITY=1        MAXPROC=2       ADEF=lhc

GROUPCFG[dzero]      FSTARGET=5      PRIORITY=100      MAXPROC=100
GROUPCFG[biome]      FSTARGET=5      PRIORITY=5        MAXPROC=2
GROUPCFG[esr]        FSTARGET=5      PRIORITY=50       MAXPROC=32      ADEF=nlgrid
GROUPCFG[ncf]        FSTARGET=40     PRIORITY=100      MAXPROC=120     ADEF=nlgrid
GROUPCFG[asci]       FSTARGET=40     PRIORITY=100      MAXPROC=120     ADEF=nlgrid
GROUPCFG[pvier]      FSTARGET=5      PRIORITY=100      MAXPROC=12      ADEF=nlgrid

# Can increase the prio for nikanter to 1000 (relative to 100 for LHC), since
# the FSWEIGHT/CREDWEIGHT is 10 as well (they will thus balance)
GROUPCFG[nikantar]   FSTARGET=2      PRIORITY=1000     MAXPROC=40      ADEF=niklocal
GROUPCFG[niktheor]   FSTARGET=3      PRIORITY=1000     MAXPROC=10      ADEF=niklocal
GROUPCFG[nikdzero]   FSTARGET=30     PRIORITY=100      MAXPROC=220     ADEF=niklocal

USERCFG[svens]       FSTARGET=0      PRIORITY=1        MAXPROC=32
USERCFG[sander]                      PRIORITY=100      MAXPROC=150     ADEF=lhc
USERCFG[s64]         FSTARGET=0      PRIORITY=1        MAXPROC=32
```

```
# versto: maxproc=120 because of size of NCF farm
USERCFG[versto]      FSTARGET=0-     PRIORITY=1        MAXPROC=120

USERCFG[davides]                     PRIORITY=500000
USERCFG[davidg]                      PRIORITY=500000
USERCFG[templon]                     PRIORITY=500000
```