Pleiadi@IRA Cluster

Technical Improvements Report

Matteo Gandolfi, Francesco Bedosti, Elia Gasperini, Matteo Stagni, Marco Tugnoli

INAF-Istituto di Radioastronomia

ABSTRACT

This technical report documents the implementation of key infrastructure improvements to the Pleiadi@IRA HPC cluster, focusing on energy efficiency and resource monitoring. The primary improvements include: (1) an automatic power management system that dynamically controls compute node power states based on workload demand, achieving a 61% reduction in idle power consumption; and (2) a comprehensive monitoring infrastructure using Slurm accounting data exported to Prometheus and visualized through Grafana dashboards.

1. INTRODUCTION

The Pleiadi@IRA cluster is a High-Performance Computing facility operated by INAF - Istituto di Radioastronomia in Bologna, Italy.

As HPC facilities face increasing pressure to reduce energy consumption while maintaining service availability, we implemented two complementary systems:

1. **Automatic Power Management:** Dynamic power control of compute nodes based on Slurm scheduler demand

2. **Resource Monitoring:** Real-time tracking of cluster utilization through Prometheus/Grafana integration

1.1 PLEIADI@IRA

PLEIADI is a project by USC-C Computing of INAF-National Institute for Astrophysics, offering high-performance computing (HPC) and high-throughput computing (HTC) resources.

Individual researchers and teams belonging to research projects, European projects, PRIN, INAF mainstream projects, scientific missions, etc. that require computing can apply requesting the resources.

The Pleiadi infrastructures is distributed on the following sites: Bologna (IRA), Catania, Trieste and Palermo. [0] [1]

Pleiadi cluster at the INAF's Istituto di Radio Astronomia (Institute of Radio Astronomy) in Bologna, also called Pleiadi@IRA, is composed of 6 chassis with 12 nodes each, for a total of 72 nodes.

The cluster is managed with Slurm scheduler and is split in 2 between "normal" Pleiadi and Pleiadi-Lofar which is dedicated to Lofar pipelines for data calibration and imaging.

The Pleiadi@IRA cluster is deeply integrated in IRA's Computing Centre and could not function stand-alone, it uses the common networking, storage, software, power and cooling of the other IRA's computing facilities.

1.2 CLUSTER INFRASTRUCTURE OVERVIEW

The cluster consists of 72 blade servers organized in 6 IBM Flex System chassis (12 blades per chassis):

Component	**Specification**
Chassis	6 × IBM Flex System Enterprise Chassis
Total compute nodes	72 (12 per chassis)
CPU per blade	2 × Intel Xeon E5-2697 v4 @ 2.30GHz (18 cores each)
RAM per blade	256 GB DDR4
Management	IPMI v2.0 via Chassis Management Module (CMM) IPMI v2.0 via (servers) Intergated Management Module (IMM)

The cluster is partitioned into two main groups: **pleiadi** (3 chassis, 36 blades) for general USC-C allocations and **pleiadi-lofar** (3 chassis, 36 blades) for dedicated LOFAR radio astronomy data processing.

The system was part of Cineca's HPC cluster Galileo, which has later been upgraded from

- Intel Haswell 2 x Intel Xeon 2630 v3 @2.4GHz 8 cores each

to

- Intel Broadwell 2 x Intel Xeon 2697 v4 @2.3GHz 18 cores each

becoming "Galileo2" [0],[1],[2]

2. Remote Power Management Infrastructure

2.1 IPMI Architecture

The cluster uses the Intelligent Platform Management Interface (IPMI) v2.0 for out-of-band management. Each one of the six chassis contains a Chassis Management Module (CMM) that provides chassis-level power monitoring, hardware health monitoring.

Each blade contains an Integrated Management Module (IMM) which provides features per-blade, for example power on and off and remote console access.

For both the CMM and IMM some features are “locked” behind the usage of proprietary IBM tools we’ll not leverage, but the features adhering the IPMI standard are enough for retrieving power consumption per-chassis and to power manage the blade servers.

2.2 IPMI Command Reference

Reading Chassis Power Consumption

The CMM provides real-time power consumption data via raw IPMI commands. The power value is returned as a 16-bit little-endian integer:

```
#!/bin/bash

# Read chassis power consumption in Watts

CMM_IP="192.168.165.130"

IPMI_USER="REDACTED"

IPMI_PASS="REDACTED"



# Raw IPMI command 0x32 0x90 returns power data

RAW_OUTPUT=$(ipmitool -I lanplus -H $CMM_IP -U $IPMI_USER \

             -P $IPMI_PASS raw 0x32 0x90 1)



# Extract bytes and convert from hex to decimal

BYTE_LOW=$(echo $RAW_OUTPUT | cut -d " " -f 5)

BYTE_HIGH=$(echo $RAW_OUTPUT | cut -d " " -f 6)

POWER_WATTS=$((16#${BYTE_HIGH}${BYTE_LOW}))



echo "Chassis power consumption: ${POWER_WATTS} W"
```

BLADE POWER CONTROL COMMANDS

The IMM allows individual blades are accessed via their dedicated IPMI interface:

```
# Check if blade is powered on or off

ipmitool -H 10.25.36.39 -U USERID -P PASSW0RD chassis power status



# Power on a specific blade

ipmitool -H 10.25.36.39 -U USERID -P PASSW0RD chassis power on



# Power off a specific blade

ipmitool -H 10.25.36.39 -U USERID -P PASSW0RD chassis power off
```

2.3 ISSUES WHILE RECONFIGURING THE CLUSTER'S IPMI

Since the system was part of Cineca's installation, all the components were configured according to their IP classes and needs.

For reconfiguring the IMMs of the single blades we were able to leverage ipmitool package which is able to talk to the IPMI from the OS, for example we can query the IPMI network config:

```
# root@r36c01s01:~# ipmitool lan print
Set in Progress         : Set Complete
Auth Type Support       : NONE MD5 PASSWORD
Auth Type Enable        : Callback :
                        : User     : MD5 PASSWORD
                        : Operator : MD5 PASSWORD
                        : Admin    : MD5
                        : OEM      :
IP Address Source       : Static Address
IP Address              : 172.18.49.101
Subnet Mask             : 255.255.0.0
MAC Address             : 40:f2:e9:c6:b6:32
SNMP Community String   : public
IP Header               : TTL=0x40 Flags=0x40 Precedence=0x00 TOS=0x10
BMC ARP Control         : ARP Responses Enabled, Gratuitous ARP Disabled
Gratituous ARP Intrvl   : 2,0 seconds
Default Gateway IP      : 172.18.255.254
Default Gateway MAC     : 00:00:00:00:00:00
Backup Gateway IP       : 0.0.0.0
Backup Gateway MAC      : 00:00:00:00:00:00
802.1q VLAN ID          : Disabled
802.1q VLAN Priority    : 0
RMCP+ Cipher Suites     : 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
Cipher Suite Priv Max   : Xaaaaaaaaaaaaaa
```

```
                          :     X=Cipher Suite Unused

                          :     c=CALLBACK

                          :     u=USER

                          :     o=OPERATOR

                          :     a=ADMIN

                          :     O=OEM

Bad Password Threshold  : 0

Invalid password disable: no

Attempt Count Reset Int.: 0

User Lockout Interval   : 0
```

The reconfiguration of the CMMs of the six chassis has been a bit more complicated since the management modules were not responding on the IPv4 addresses and needed to be contacted on the IPv6 local-link addresses.

Then we were able to access the management web interface and change the network configurations.

This table recaps the network changes:

chassis	name	mac	old ipv4	ipv4	ipv6_LL
1	r036c01fpc-mgt	6c:ae:8b:5f:7a:19	10.26.36.103	192.168.165.130	fe80::6eae:8bff:fe5f:7a19(/64)
2	r036c02fpc-mgt	6c:ae:8b:5f:79:ba	10.26.36.109	192.168.165.131	fe80::6eae:8bff:fe5f:79ba
3	r036c03fpc-mgt	6c:ae:8b:5f:7a:25	10.26.36.115	192.168.165.132	fe80::6eae:8bff:fe5f:7a25
4	r036c04fpc-mgt	6c:ae:8b:5e:23:6a	10.26.36.123	192.168.165.133	fe80::6eae:8bff:fe5e:236a
5	r036c05fpc-mgt	6c:ae:8b:5f:79:e5	10.26.36.129	192.168.165.134	fe80::6eae:8bff:fe5f:79e5
6	r036c06fpc-mgt	6c:ae:8b:5e:e0:50	10.26.36.135	192.168.165.135	fe80::6eae:8bff:fe5e:e050

3. Automatic Power Management System

3.1 Overview

The automatic power management system was activated in April 2024 to reduce energy consumption during periods of low cluster utilization. The system integrates with the Slurm workload manager to dynamically power nodes on and off based on job queue demand: when the queued jobs do not require a node, it enters an "idle" state and is automatically shut down.

Conversely, when new jobs are submitted and additional resources are required, the system powers the needed nodes back on and waits for them to rejoin the cluster before dispatching the workload.

This mechanism allows the cluster to scale its active size in response to actual usage, significantly reducing power draw and heat generation during low-activity periods without requiring manual intervention.

It can also be used to manually cap the number of active nodes when needed: for example, if the cooling system is experiencing issues and heat generation needs to be reduced.

3.2 Configuration

Node classification for power management:

Category	Count	Nodes	Purpose
Always On	8	`r36c[01-04]s[01-02]`	Interactive sessions for both pleiadi and pleiadi-lofar (4 nodes each)
Power-Managed	64	Remaining blades	Dynamic power control

Slurm Configuration

The power management is implemented via Slurm's `SuspendProgram` and `ResumeProgram` directives in `slurm.conf`:

```
# /etc/slurm/slurm.conf (excerpt)

SuspendProgram=/usr/local/bin/nodesuspend

ResumeProgram=/usr/local/bin/noderesume

ResumeFailProgram=/usr/local/bin/nodefailresume

SuspendExcNodes=r36c[01-04]s[01-02]

SuspendTimeout=120

ResumeTimeout=900
```

```
SuspendRate=10

ResumeRate=10

DebugFlags=Power,NO_CONF_HASH

TreeWidth=1000

#see: https://bugs.schedmd.com/show_bug.cgi?id=14270

PrivateData=cloud

SuspendTime=7200
```

Notes: we are not leveraging some configuration options since as of 2025-08 our Slurm version is < 23.0 since the whole computing centre is still running on Debian 11.

With the upcoming upgrade to Debian version 13, Slurm will be upgraded to version 24.x.

For example this config line shoul allow us to keep “n” nodes from suspension:

```
SuspendExcNodes=r36c[05]s[09-12]:n
```

Other configurations not working on versions < 23.x:

```
SuspendExcStates=

ReconfigFlags=KeepPowerSaveSettings
```

POWER CONTROL SCRIPTS

It’s a series of scripts written by Ole Holm Nielsen [3] that allow power management of nodes in Slurm:

- **nodesuspend** is called by slurmctld via SuspendProgram when nodes become idle long enough to be powered down. It queries each node's Slurm feature tags and dispatches the appropriate shutdown method: IPMI for physical nodes (power_ipmi), Azure deallocation for cloud VMs (power_azure), or a no-op for testing (power_noaction). The script handles heterogeneous clusters by routing each node to the right backend based on its feature tag in slurm.conf.
- **noderesume** is called via ResumeProgram when new jobs need nodes that are currently powered off. It does the same feature/tag lookup and dispatches the appropriate power-on method (power_ipmi to boot physical nodes via IPMI, power_azure to start cloud VMs). Slurm then waits for the nodes to check in before assigning jobs.

- **nodefailresume** is called via ResumeFailProgram when a node fails to come back online within Slurm's ResumeTimeout. It logs the failure to /var/log/slurm/nodefailresume.log and emails the sysadmin with the output of sinfo -lRN (the list of down/drained nodes with their reason), so the admin has immediate context to investigate what went wrong.
- **power_ipmi** is a helper script that performs the actual IPMI power commands against physical nodes via their BMCs. It accepts a flag for the desired action (-r to power on, -s for graceful shutdown, -f for forced off, -c for cycle, -q to query status) and a Slurm nodelist. It constructs the BMC hostnames by appending a suffix or a prefix to each node name, reads IPMI credentials from the slurm user's .bashrc, then calls FreeIPMI's ipmipower against all BMCs in a single consolidated call. All actions except query are logged to /var/log/slurm/power_ipmi.log.

Suspend Script (`/usr/local/bin/nodesuspend`):

```
#  !/usr/bin/env bash


# Slurm nodes suspend/resume script

# Helper scripts required: power_ipmi power_azure power_noaction


# We must define some node features power_xxx in slurm.conf, for example for IPMI
and Azure cloud:

# NodeName=node[001-100] Feature=xeon2650v4,opa,xeon24,power_ipmi

# NodeName=cloud[001-100] Feature=xeon8272cl,power_azure

# We can read the node features using:

#   sinfo -hN -O "Nodelist: ,Features:" --nodes=$* | uniq


# Define the action

action="-s"


# The environment variables passed from slurmctld do not include USER etc.

export USER=`whoami`

export PATH=/usr/local/bin:$PATH


# Check for empty argument list

if [[ $# < 1 ]]
```

```
then
   exit 0
fi

# Get the node list including features (in a single call to minimize load on slurmctld)
TMPFILE=`mktemp`
sinfo -hN -O "Nodelist: ,Features:" --nodes=$* > $TMPFILE

# We require the "nodeset" command from the ClusterShell package.  Install it by:
# yum install epel-release
# yum install clustershell

# Node suspend/resume DUMMY action used only for testing the power_save module
# Select the nodelist with the "power_noaction" feature
nodelist=`grep power_noaction $TMPFILE | uniq | awk '{print $1}' | nodeset --fold`
if [[ -n "$nodelist" ]]
then
   power_noaction $action $nodelist
fi

# Node suspend/resume by IPMI
# Select the nodelist with the "power_ipmi" feature
nodelist=`grep power_ipmi $TMPFILE | uniq | awk '{print $1}' | nodeset --fold`
if [[ -n "$nodelist" ]]
then
   power_ipmi $action $nodelist
fi

# Execute Azure VM nodes suspend/deallocate
# Select the nodelist with the "power_azure" feature
```

```
nodelist=`grep power_azure $TMPFILE | uniq | awk '{print $1}' | nodeset --fold`

if [[ -n "$nodelist" ]]

then

   power_azure $action $nodelist

fi



# Cleanup

rm -f $TMPFILE
```

Resume Script (`/usr/local/bin/noderesume`):

```
#  !/usr/bin/env bash



# Slurm nodes suspend/resume script

# Helper scripts required: power_ipmi power_azure power_noaction



# We must define some node features power_xxx in slurm.conf, for example for IPMI
and Azure cloud:

# NodeName=node[001-100] Feature=xeon2650v4,opa,xeon24,power_ipmi

# NodeName=cloud[001-100] Feature=xeon8272cl,power_azure

# We can read the node features using:

#   sinfo -hN -O "Nodelist: ,Features:" --nodes=$* | uniq



# Define the action

action="-r"



# The environment variables passed from slurmctld do not include USER etc.

export USER=`whoami`

export PATH=/usr/local/bin:$PATH



# Check for empty argument list

if [[ $# < 1 ]]
```

```
then
   exit 0
fi

# Get the node list including features (in a single call to minimize load on slurmctld)
TMPFILE=`mktemp`
sinfo -hN -O "Nodelist: ,Features:" --nodes=$* > $TMPFILE

# We require the "nodeset" command from the ClusterShell package.  Install it by:
# yum install epel-release
# yum install clustershell

# Node suspend/resume DUMMY action used only for testing the power_save module
# Select the nodelist with the "power_noaction" feature
nodelist=`grep power_noaction $TMPFILE | uniq | awk '{print $1}' | nodeset --fold`
if [[ -n "$nodelist" ]]
then
   power_noaction $action $nodelist
fi

# Node suspend/resume by IPMI
# Select the nodelist with the "power_ipmi" feature
nodelist=`grep power_ipmi $TMPFILE | uniq | awk '{print $1}' | nodeset --fold`
if [[ -n "$nodelist" ]]
then
   power_ipmi $action $nodelist
fi

# Execute Azure VM nodes suspend/deallocate
# Select the nodelist with the "power_azure" feature
```

```
nodelist=`grep power_azure $TMPFILE | uniq | awk '{print $1}' | nodeset --fold`
if [[ -n "$nodelist" ]]
then
   power_azure $action $nodelist
fi

# Cleanup
rm -f $TMPFILE
```

Failresume Script (`/usr/local/bin/nodefailresume`):

```
#!/usr/bin/env bash

# nodefailresume script

# The environment variables from slurmctld do not include USER etc.
export USER=`whoami`
export PATH=/usr/local/bin:$PATH
# Logfile for suspend/resume actions
# N.B.: Make sure this file is writable by user slurm
export LOGFILE=/var/log/slurm/nodefailresume.log
# Make sure the LOGFILE has correct permissions
touch $LOGFILE
chown slurm: $LOGFILE
chmod 644 $LOGFILE

export DATE=`date +"%b %d %T"`
echo "$DATE nodefailresume nodes $@" >> $LOGFILE

slurm_notify=<sysadmin-email>
my_mail=/usr/bin/mailx
```

```
# Notify Slurm administrator of failed node resume
sinfo -lRN | $my_mail -s "Nodes $@ failed to resume" $slurm_notify
```

Helper scripts: power_ipmi Script (`/usr/local/bin/power_ipmi`):

```
#!/usr/bin/env bash

# Suspend/resume IPMI-based Slurm nodes using FreeIPMI tools from https://www.gnu.org/software/freeipmi/
# Author: Ole.H.Nielsen@fysik.dtu.dk
# Homepage: https://github.com/OleHolmNielsen/Slurm_tools/

# Use with ResumeProgram and SuspendProgram in slurm.conf
# NOTE: The slurmctld will execute this script as user "slurm"
# (see https://slurm.schedmd.com/power_save.html)
# so the slurm user must have credentials for suspending and resuming nodes.

# MODIFY THIS:
# Add these lines (uncommented) to the users' .bashrc file which should export variables like:
export IPMI_USER=REDACTED
export IPMI_PASSWORD= REDACTED
# Define the node BMC DNS name: BMC DNS-name is the node name plus this suffix:
BMC_SUFFIX="b"
BMC_PREFIX="mng-"
# For example: node c190 BMC has DNS name c190b

# Logfile for IPMI suspend/resume actions
# NOTE: Make sure this file is writable by SlurmUser
```

```
LOGFILE=/var/log/slurm/power_ipmi.log

# The slurm user must own the $LOGFILE

slurmuser="`scontrol show config | grep SlurmUser | awk '{split($3,a,"("); print
a[1]}'`"



# Prerequisites:

# * Install this RPM package: yum install freeipmi

# * We require the "nodeset" command from the ClusterShell package.  Install it by:

#   yum install epel-release

#   yum install clustershell



# Command usage:

function usage()

{

cat <<EOF

Usage: $0 [-r|-s|-q|-h] nodelist

where the action is:

   -r: Resume (start) nodes in nodelist

   -s: Suspend (stop) nodes in nodelist

   -c: Power cycle nodes in nodelist

   -q: Query power status of nodes in nodelist

   -h: Print this help information

EOF

}



# Set the ipmipower command action

action=""

logging=1

while getopts "rscqh" options; do

   case $options in
```

```
          r )    action="--on --on-if-off"
                 ;;
          s )    action="--soft --wait-until-off"
                 ;;
          c )    action="--cycle"
                 ;;
          q )    action="--stat"
                 logging=0
                 ;;
          h|? ) usage
                 exit 0;;
          * ) usage
                 exit 1;;
   esac
done
shift $((OPTIND-1))

# Check the Slurm nodelist
if [[ $# != 1 ]]
then
   echo "ERROR: No Slurm nodelist has been given"
   usage
   exit 1
fi

# List of nodenames and BMC DNS names
nodelist=$1
nodecount=`nodeset --count $nodelist`
# Append the BMC's DNS name suffix BMC_SUFFIX to the nodes' DNS names
# using the ClusterShell command "nodeset"
```

```
bmclong=`nodeset -O "${BMC_PREFIX}%s" --expand $1`

bmclist=`nodeset --fold $bmclong`



# Source the users' .bashrc file which should export variables like:

# export IPMI_USER=root

# export IPMI_PASSWORD=verysecretpassword

# Note: The environment variables set by slurmctld do NOT include PATH, USER, HOME
etc.

source ~/.bashrc

USER=`whoami`



# Prepend the path where FreeIPMI tools live

export PATH=/usr/sbin:$PATH



if [[ -z "$IPMI_USER" || -z "$IPMI_PASSWORD" ]]

then

   echo "ERROR: The user IPMI_USER and/or password IPMI_PASSWORD have not been set
in ~/.bashrc"

   exit 1

fi



if [[ -z "$action" ]]

then

   echo "ERROR: No action has been given"

   usage

   exit 1

elif [[ $logging -eq 1 ]]

then

   # The case where we want logging to go to $LOGFILE

   # Make sure the LOGFILE is owned by SlurmUser and has correct permissions

   touch $LOGFILE
```

```
   chown $slurmuser: $LOGFILE

   chmod 644 $LOGFILE

   # Do the resume or suspend action:

   # Redirect stdout and stderr to $LOGFILE

   exec &>> $LOGFILE

   DATE=`date +'%b %d %T'`

   echo "$DATE Invoked $0" by `id $USER`

   # Display $action in UPPER case (see the bash man-page under Case modification)

   echo "$DATE POWER ${action^^} the IPMI based nodelist $nodelist ($nodecount
nodes)"

fi



#

# Use The FreeIPMI command "ipmipower" to power nodes on or off or get status

#

# Specify the IPMI 2.0 cipher suite ID to use:

# HPE and SuperMicro BMC only support "-I 3"

#cipher="-I 17"

cipher=""

# fallbackcipher="-I 3"

fallbackcipher=""

driver="-D LAN_2_0"

timeout="--session-timeout=1000"     # Set IPMI timeout to 1000 milliseconds

consolidate="--consolidate-output"   # Consolidated node output



# First try cipher suite 17 with a fallback to 3 (for HPE and SuperMicro BMCs)

tempfile=`mktemp`

ipmipower $driver $cipher --username=$IPMI_USER --password=$IPMI_PASSWORD $timeout
$consolidate --hostname=$bmclist $action > $tempfile 2>&1

if [ $? -eq 0 ]

then
```

```
   cat $tempfile

else

   cat $tempfile

   ipmipower      $driver      $fallbackcipher      --username=$IPMI_USER      --
password=$IPMI_PASSWORD $timeout $consolidate --hostname=$bmclist $action

fi

rm -f $tempfile
```

Usage example: query power status manually:

```
root@scheduler:~# power_ipmi -q r36c[01]s[01-12]



----------------

mng-r36c01s[01-10]

----------------

 on



----------------

mng-r36c01s[11-12]

----------------

 off
```

3.3 Power Consumption Results

Measurements per chassis under different load conditions:

State	Power (W)	Description
All blades ON (idle)	~860	12 blades, no compute load
All blades OFF	~220	CMM and infrastructure only
2 blades ON	~360	Minimum always-on config

Total Rack Power Savings: Reduced from ~4.6 kW to ~1.8 kW (**61% reduction**) when idle.

4. Resource Monitoring with Slurm Accounting

4.1 Architecture Overview

The monitoring system exports Slurm accounting data to Prometheus, enabling visualization through Grafana dashboards.

Prometheus is an open-source monitoring toolkit that collects and stores metrics as time series data; node_exporter is a Prometheus agent that runs on the host and exposes system metrics — its textfile_collector extension allows custom metrics written to a file to be picked up automatically; Grafana is an open-source visualization platform that queries Prometheus and displays the data as interactive dashboards.

The data flow is: Slurm Scheduler → Export Script → node_exporter (textfile_collector) → Prometheus → Grafana.

4.2 Slurm Accounting Data

The `sshare` command provides cumulative usage statistics for all accounts:

```
# Query Slurm accounting data

sshare -a -o User,Account,RawUsage -n --parsable



# Example output:

# |lofar|5423456789|

# fdg|lofar|4532100000|

# |pleiadi|2345678901|
```

```
# pleia12|pleiadi|1234567890|
```

4.3 PROMETHEUS EXPORT SCRIPT

The following script runs periodically via cron to export Slurm data to Prometheus format (`/iranet/sys/iratoolz/slurm_accounting.sh`):

```
#!/usr/bin/env bash



#for i in $(sshare -a -o User,Account,RawUsage -n --parsable); do

#    echo "$i"

#done



sshare -a -o User,Account,RawUsage -n --parsable > /tmp/corehours.txt



echo       "#HELP       slurm_acct_users       Slurm       acct       users"       >
/var/lib/node_exporter/textfile_collector/slurm_acct_users.prom

echo             "#TYPE             slurm_acct_users             gauge"             >>
/var/lib/node_exporter/textfile_collector/slurm_acct_users.prom



echo       "#HELP       slurm_acct_groups       Slurm       acct       groups"       >
/var/lib/node_exporter/textfile_collector/slurm_acct_groups.prom

echo             "#TYPE             slurm_acct_groups             gauge"             >>
/var/lib/node_exporter/textfile_collector/slurm_acct_groups.prom







while read p; do

#  echo "$p"



type=""

name=""
```

```
group=""

value=""




  if [[ $p = '|'* ]]

  then

#funziona ma lo ho disabilitato: se il gruppo e' root, skippa

#   if [[ $(echo "$p" | tr -d ' ' | cut -d '|' -f2) = "root" ]]

#   then

#          continue

#   else

           type="group"

           name=$(echo "$p" | tr -d ' ' | cut -d '|' -f2)

            value=$(echo "$p" | tr -d ' ' | cut -d '|' -f3)

           echo               "slurm_acct_groups{name=\"$name\"}               $value"
>>/var/lib/node_exporter/textfile_collector/slurm_acct_groups.prom

#   fi

  else

    type="user"

    name=$(echo "$p" | tr -d ' ' | cut -d '|' -f1)

    group=$(echo "$p" | tr -d ' ' | cut -d '|' -f2)

    value=$(echo "$p" | tr -d ' ' | cut -d '|' -f3)

    echo      "slurm_acct_users{name=\"$name\",group=\"$group\"}      $value"      >>
/var/lib/node_exporter/textfile_collector/slurm_acct_users.prom

  fi



#echo "- - -"


done < /tmp/corehours.txt
```

```
rm -f /tmp/corehours.txt




##HELP slurm_acct_users Slurm acct users

##TYPE slurm_acct_users gauge

#slurm_acct_users{name="$p",group="$group",type="$type"}

#

##HELP slurm_acct_groups Slurm acct groups

##TYPE slurm_acct_groups gauge

#slurm_acct_groups{name="$p",type="$type"}



## HELP node_systemd_unit_state Systemd unit

## TYPE node_systemd_unit_state gauge

#node_systemd_unit_state{name="ModemManager.service",state="activating",type="dbus"
} 0

#node_systemd_unit_state{name="ModemManager.service",state="active",type="dbus"} 1
```

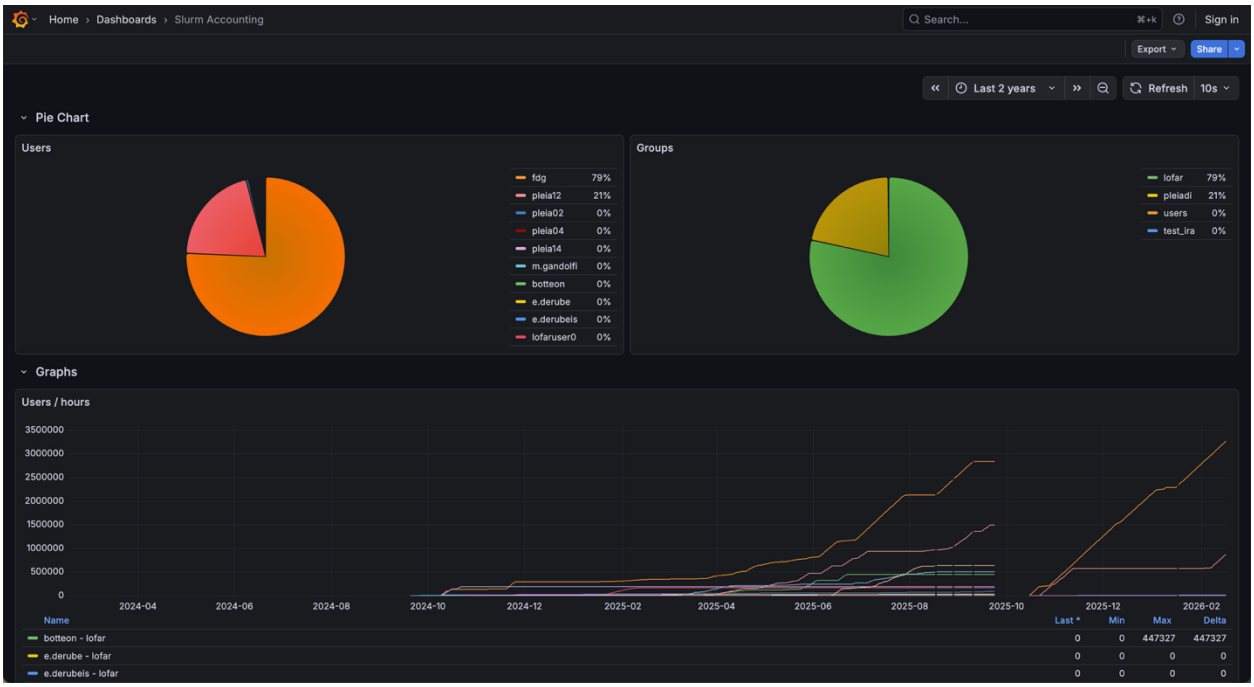
Cron configuration (`/etc/cron.d/slurm-accounting-export`):

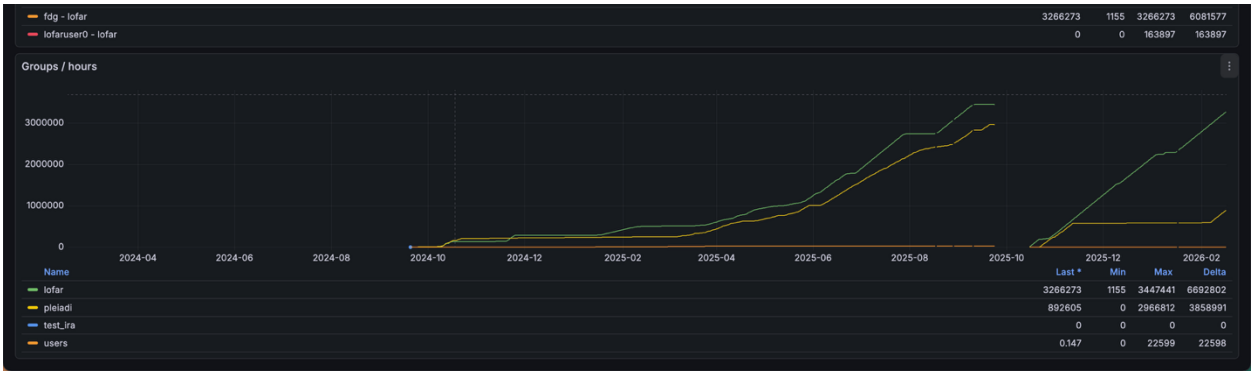
```
# Export Slurm accounting to Prometheus every 5 minutes

*5 *  * * *   root    /iranet/sys/iratoolz/slurm_accounting.sh  > /dev/null 2>&1
```

4.4 GRAFANA DASHBOARD QUERIES

Example of the production dashboard:





4.5 Issues

Due to an unforseen slurmdb incompatibility during our version upgrade from Debian 11 to 13 we had to start over with the database, this is the reason on 2025-10 the usage has been reset.

This has not been considered a blocking issue since the new call started on 2025-09 and the interesting data is the per-user delta between the start and the end of the call allocation.

5. Results and Impact

5.1 Direct Energy Savings

The automatic power management system achieved significant energy reduction:

Metric	Before	After	Improvement
Idle Power	4.6 kW	1.8 kW	**61% reduction**
Annual Energy (est.)	40,296 kWh	15,768 kWh	~24,500 kWh saved

5.2 Indirect Energy Savings

The automatic shutdown of the un-needed nodes also keeps the heat production to a minimum thus stressing less the cooling systems of the computing centre.

NB: this benefit does not give more headroom regarding the sizing of the cooling system since the heat production when running all the nodes is the same.

5.3 Monitoring Benefits

- Real-time visibility into cluster utilization
- Historical analysis for capacity planning
- Per-user accounting for resource allocation fairness
- Data-aided reporting for USC-C calls

6. Conclusions

The infrastructure improvements to the Pleiadi@IRA cluster have achieved:

1. 61% reduction in idle power consumption through automatic node power management

2. Comprehensive resource monitoring via Prometheus/Grafana integration

3. Operational documentation for ongoing maintenance and troubleshooting

These improvements support INAF's commitment to sustainable HPC operations while maintaining high availability for scientific research workloads.

7. Sources/Links:

- https://pubs.lenovo.com/sd650/ipmi_command
- https://lenovopress.lenovo.com/sg248152.pdf
- https://github.com/OleHolmNielsen/Slurm_tools/tree/master/power_save
- https://slurm.schedmd.com/SLUG23/DTU-SLUG23.pdf
- https://wiki.fysik.dtu.dk/Niflheim_system/Slurm_cloud_bursting/#configuring-slurm-conf-for-power-saving
- [0]: https://en.wikipedia.org/wiki/Galileo_(supercomputer)
- [1]: https://www.hpc.cineca.it/systems/hardware/
- [2]: https://www.hpc.cineca.it/newsletter/new-galileo-in-production-again/
- [3]: https://github.com/OleHolmNielsen/Slurm_tools/tree/master/power_save