Casper Slurm User Manual

Staff of HPC@Polito -

This guide is written for all users of our clusters
to answer all frequently asked questions about the usage of HPC systems
If you need more information
don't hesitate to contact us at
hpc.dauin@polito.it

Contents

Intended Audience

This guide will show the usage of the HPC@POLITO system that are running SLURM as a scheduler, right now CASPER and in the near future HACTAR. The first four section of this document can be used for both the cluster, is enough substitute to {login-node} the right address of the respetive login node of the cluster you want to use.

[Casper] *casperlogin.polito.it*

[Hactar] *hactarlogin.polito.it*

If you are familiar with SGE scheduler here a list of thing to keep in mind when using SLURM:

1. SLURM does not have parallel environments (orte, shared or anything else), you will use the directives `--nodes` and `--tasks-per-node` to specify the distribution of the resources that you need.
2. The SGE queue are called partition in SLURM
3. SGE let you chose hard limit and virtual free memory size, instead in SLURM you specify parameters for memory (`--mem` and `--mem-per-cpu`), which is the limit for your real memory usage and drives the decision where your job is started. If you don't know how many memory your program will use let the scheduler uses its default value.
4. Some environment variables may have changed (Ex. SGE `$JOB_ID` in SLURM become `$SLURM_JOB_ID` or SGE `$NSLOTS` in SLURM become `$SLURM_NTASKS`), for a complete reference you could read https://slurm.schedmd.com/sbatch.html.

Gentlemen Agreements

- By using our systems for research purpose, you automatically authorize HPC@POLITO's staff to publish your personal data (name, surname, research group) and data associated with your research on our web site (hpc.polito.it) and in all the other papers published by HPC@POLITO (annual reports, presentations, etc.).
- By using our systems for research purpose, you agree to quote the HPC@POLITO's center in all your scientific articles on paper, conference or books. We kindly suggest this kind of acknowledgement:

  "Computational resources provided by HPC@POLITO, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it)".

  Or a shorter version as
  "Computational resources provided by HPC@POLITO (http://www.hpc.polito.it)".

Rules for Cluster Uses

- It is NOT permitted to run commands like simulations and computations directly from the command line, due to existing risk to make unusable *login-node* or other shared resources (by running such commands your account will be blocked). You MUST always pass through scheduler's computational queues (also for compilation purpose).
- Any uses of shared resources except didactical or research purposes are NOT permitted.
- Every user is responsible for activity done by his account, it is not recommended to share your account credentials with others, it is just allowed by previously informing HPC staff;

  Generally, it's NOT permitted to share sensible data, especially username and password, with others. The user agree to hold them safely.
- As a general rule, it is NOT permitted to do something which is not clearly permitted.

This document contains only technical and practical information and must be considered only as a supplement for the complete HPC@POLITO regulation.
Updated versions of regulation or this technical manual can be reached on the project's website: *http://hpc.polito.it*.
All the users must take vision of the regulation and comply with it.

HPC@POLITO Academic Computing Center

Politecnico di Torino HPC project is an Academic Computing center which provides computational resources and technical support for research activities in accademic and didattical purposes.
The HPC project is officially managed by LABINF (Laboratorio Didattico di Informatica Avanzata) under supervision of DAUIN (Department of Control and Computer Engineering) which granted by Board of Directors.

Casper technical specification

Architecture	Linux Infiniband-DDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband DDR 20 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Opteron 6276/6376 (Bulldozer) 2.3 GHz (turbo 3.0 GHz) 16 cores
Performance	4.360 TFLOPS
Power Consumption	7 kW
Computing Cores	512
Number of Nodes	16
Total RAM Memory	2.0 TB DDR3 REGISTERED ECC
OS	Centos 7.4 - OpenHPC 1.3.3
Scheduler	SLURM 17.02.9

Hactar technical specification

Architecture	Linux Infiniband-QDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband QDR 40 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Xeon E5-2680 v3 2.50 GHz (turbo 3.3 GHz) 12 cores
GPU Node	2x Tesla K40 - 12 GB - 2880 cuda cores
Performance	9.7 TFLOPS (July 2015)
Computing Cores	696
Number of Nodes	29
Total RAM Memory	3.7 TB DDR4 REGISTERED ECC
OS	CentOS 6.6
Scheduler	GridEngine 2011.11

Storages technical specification

Home Storage	140 TB on RAID 6, throughput near 200 MB/s
Lustre Storage	87 TB. throughput greater then 2.2 GB/s
Storage Interconnect	Ethernet 10 Gb/s

First Access

Your account has to be considered active from the moment you received confimation email containing your access credentials.

How get access to clusters? It depends on which operating system is used by your computer. Assume that you are using Linux, Unix or OSX system, you can use a simple ssh client from any terminal with following:

$ ssh username@{login-node}

to access on our systems please enter username and password associated to you by means of confirmation email.

If you are using a Windows system we suggest you to use PuTTY software (available at http://www.putty.org) by setting the configurations as following:

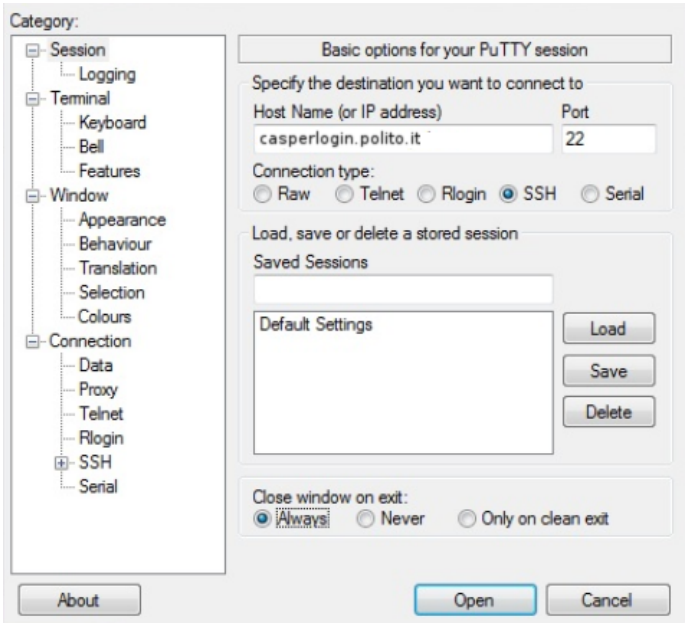

Figure 1: Putty - *PuTTY* Setting for access by Windows OS

We strongly suggest you to change your password after first log-in by usign command following:

$ passwd

There is no limitation about length and composition of password, in any case is suggested to choose a password which is a combination of at least 8 characters, numbers, uppercase and lowercase letters.

Let us remind you that account sharing (the sharing of the same account among different users) is not permitted, moreover the person who made the request for the account has to take all the responsibility to keep the credentials safe: choose a strong password, don't share your sensible data with others. Your account should be secure, in any case if you observe unexpected account behaviors, please contact HPC's Staff immediately.

File Transfer & Storage

After the access stage, the first showed directory is the user's *home*, located in the *Home Storage* and accessible by both clusters:

Home storage

/home/username/

where the data have to be put in order to start a task and where the data will be written at the end of each task.
This directory can be accessed only by the owning user. Remember that in this storage each user can store at most 1 TB of data.
To copy files/directories inside your main directory you can use *scp* command as following:

$ scp -r /path/to/local/dir/ username@{login-node}:/home/username

Instead, to copy files from a CLUSTER on your local machine, use command following:

$ scp -r username@{login-node}:/home/username /path/to/local/dir/

It works similar as the *cp* command in Unix environment.

$ cp SOURCE DEST

Third option is to use the SFTP server which is available within the cluster.
For example, you can use the FileZilla software configured as following.

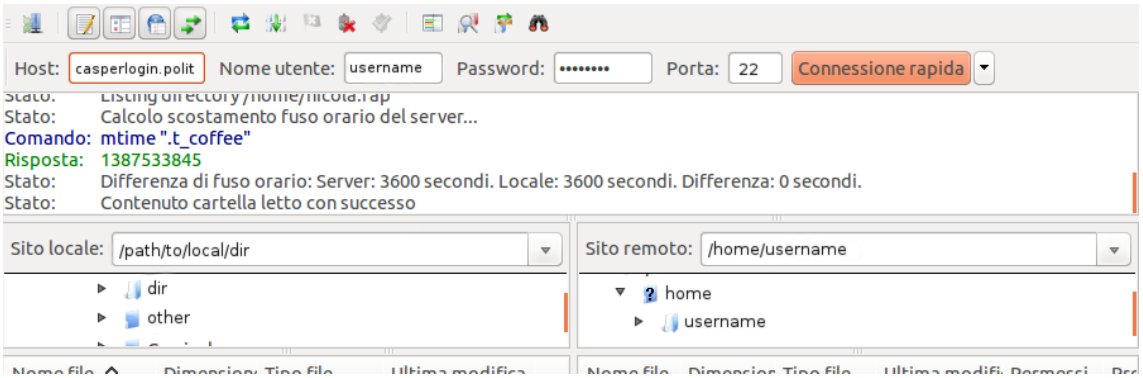

Figure 2: FileZilla configuration for file transfer on {login-node}

High performance Storage

All users can access an additional high performance Storage. It is provided with Lustre filesystem and it will improve all the tasks that make massive usage of I/O on files.

This storage is accessible by this path:

/work

This is as a secondary *home directory* containing for each user has a directory (named with his self username) and permissions to operate on it.

In order to move files from the working storage to the Lustre storage, you can use the *cp* command syntax as:

cp /home/user_name/file_name /work/user_name/file_name

cp -r /home/user_name/directory_name /work/user_name/directory_name

On this storage unit each user can wholly access 1TB of space (*soft quota*), but if strictly necessary it is possible to pass this threshold until 1.5TB (*hard quota*) for a short period (7 days).
To check your quota, use:

$ get-my-lustre-quota -h

The storage capacity is about 87TB and can reach a performance 1.1 GB/s used from CASPER and more than 2.2 GB/s from HACTAR.
Each user can send a request by mail to obtain more space (extend their quota). The mail have to be sent to *hpc.dauin@polito.it* and contain a short explanation; The request will be evaluated by HPC@POLITO Staff.

Local Storage on Nodes

For each jobs it is possible to temporary use the additional space present on the local disk of the computational nodes.
This space can be accesses by the following path:

/scratch 1TB

inside the job is possible to refer to a specific temporary folder automatically created when the job is started on the node ad deleted at the end of the job. Environment variable $TMPDIR will contain the path to the temporary job folder.
PAY ATTENTION:

1. - the space availability depend on the other job usage of the resource
2. - is deprecated the usage of the path /scratch, use instead the automatic handled dir in the path `$TMPDIR`

Cluster Usage

The runnable processes on a cluster are "batch processes"; it means that are not interactive and execution of them can be postponed. Each task/job is composed by one or more processes that work together to achieve a certain result.
A job is executed after his schedulation; in order to schedule a job it must be inserted in a waiting list, managed by cluster, waiting to get the available resources held by oldest processes.
CASPER uses the SLURM scheduler to manage the waiting queues and availability of the computational shared resources.

The public partition for submitting the job is:

global	Default partition for public access, which includes all nodes; the maximum duration of a job is 10 days.

Control and Submission of a Job

sbatch

All the jobs should be passed to the cluster scheduler through submission. It is possible to submit jobs using *sbatch* command which recevices as argument a reference to a script file containing all the required information.
The script file, know as *sbatch script*, is mainly composed by a set of directives and an execution command as should be appear on a command line.

An example of this file is showed following:
All the directive for the scheduler are the one that starts with `#SBATCH`

[script.sbatch]

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@polito.it
#SBATCH --partition=global
#SBATCH --time=hh:mm:ss
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --output=job_name_%j.log
#SBATCH --mem-per-cpu=1024M
example_task.bin
```

Most Important Directives:

`--output`	standard output directly to the file name specified, default both standard output and standard error are directed to the same file.
`--error`	instruct Slurm to connect the batch script's standard error directly to the file name specified.
`--mail-user`	Email where the information on task will be sent.
`--mail-type`	Events which cause an email notification (*es. ALL*). Valid type values are NONE, BEGIN, END, FAIL, REQUEUE, ALL.
`--workdir=`{directory}	If present cause the execution of task using the {directory} as working directory (the path of {directory} can be specified as full path or relative path).
`--ntasks-per-node`	number of tasks per node, if used with the `--ntasks` option, the `--ntasks` option will take precedence.
`--ntasks`	Total number of tasks for the job.
`--nodes`	Number of nodes that will be used.
`--time`	Indicates the hard run time limit, which is about the time that processes needs to reach the end.This value must be less than 10 days (<240 hours) for tasks on *all.q* queue.
`--mem-per-cpu`	Minimum memory required per allocated CPU, default units are megabytes (default=1000).
`--mem`	Specify the real memory required per node, default units are megabytes.
`--partition`	Indicates the partition where the job has to be scheduled (default=global).
`--exclude`	Explicitly exclude certain nodes from the resources granted to the job.
`--constraint`	Nodes have features assigned to them and users can specify which of these features are required by their job using the constraint option. Defined constraint are: amd6376, amd6276, amd6274, amd6128.

For a complete reference you could read https://slurm.schedmd.com/sbatch.html.

squeue, sinfo, sjstat, scancel, sprio

In order to get information on the scheduled jobs with *sbatch*, as *job-ID*, *status*, *partition* and the used slots number, it's possible to use command following:

```
[casper]$ squeue -u username

      JOBID PARTITION     NAME     USER   ST       TIME     NODES  NODELIST(REASON)
        600    global     test   username  R   2-23:29:19      1   compute-0-1
        601    global     test   username  R      3:53:04      3   compute-0-[2-4]
        602    global     test   username  R         7:55      1   compute-0-8
```

To obtain more information about state of each specific job you can use:

```
$ scontrol show job 600
JobId=600 JobName=test
   UserId=test(500) GroupId=test(500) MCS_label=N/A
   Priority=30937 Nice=0 Account=test QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=04:07:05 TimeLimit=8-08:00:00 TimeMin=N/A
   SubmitTime=2018-02-13T10:25:11 EligibleTime=2018-02-13T10:25:11
   StartTime=2018-02-13T10:25:12 EndTime=2018-02-21T18:25:12 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=global AllocNode:Sid=casperlogin:30786
   ReqNodeList=(null) ExcNodeList=compute-0-8
   NodeList=compute-0-[2-4]
   BatchHost=compute-0-2
   NumNodes=3 NumCPUs=96 NumTasks=96 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=96,mem=240G,node=3
   Socks/Node=* NtasksPerN:B:S:C=32:0:*:* CoreSpec=*
   MinCPUsNode=32 MinMemoryNode=40G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/test/test.bin
   WorkDir=/home/test/
   StdErr=/home/test/job_600.log
   StdIn=/dev/null
   StdOut=/home/test/job_600.log
   Power=
```

To obtain information about the entire cluster and the jobs executing on it, use:

```
[casper]$ squeue
```

Partition status can be checked by:

```
[casper]$ sinfo

PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
global*        up 10-00:00:0      1    mix compute-0-1
global*        up 10-00:00:0      7  alloc compute-0-[2-8]
global*        up 10-00:00:0      8   idle compute-0-[9-16]
lisa           up   infinite      1  alloc compute-0-5
bioeda         up   infinite      3  alloc compute-0-[6-8]
nqs            up   infinite      3   idle compute-0-[9-11]
modena         up   infinite      2   idle compute-0-[12-13]
pt-erc         up   infinite      3   idle compute-0-[14-16]
maintenance    up   infinite      1    mix compute-0-1
maintenance    up   infinite      7  alloc compute-0-[2-8]
maintenance    up   infinite      8   idle compute-0-[9-16]
```

or using:

```
[casper]$ sjstat

Scheduling pool data:
-------------------------------------------------------------
Pool        Memory  Cpus  Total Usable   Free  Other Traits
-------------------------------------------------------------
global*    128752Mb    32      4      4      0  local,public,amd6276
global*    128752Mb    32      3      3      0  local,private,amd6274
global*    128752Mb    32      1      1      1  local,private,amd6128
global*    128752Mb    32      3      3      3  local,private,amd6276
global*    128752Mb    32      5      5      5  local,private,amd6376
lisa       128752Mb    32      1      1      0  local,private,amd6274
bioeda     128752Mb    32      2      2      0  local,private,amd6274
bioeda     128752Mb    32      1      1      1  local,private,amd6128
nqs        128752Mb    32      3      3      3  local,private,amd6276
modena     128752Mb    32      2      2      2  local,private,amd6376
pt-erc     128752Mb    32      3      3      3  local,private,amd6376
maintenan  128752Mb    32      4      4      0  local,public,amd6276
maintenan  128752Mb    32      3      3      0  local,private,amd6274
maintenan  128752Mb    32      1      1      1  local,private,amd6128
maintenan  128752Mb    32      3      3      3  local,private,amd6276
maintenan  128752Mb    32      5      5      5  local,private,amd6376

Running job data:
---------------------------------------------------------------------------
JobID    User      Procs Pool      Status        Used  Master/Other
---------------------------------------------------------------------------
```

To view the components of all the jobs scheduling priority use:

```
[casper]$ sprio

          JOBID   PRIORITY        AGE  FAIRSHARE    JOBSIZE  PARTITION
```

usefull Option:

`--jobs={jobID}`	Print the job priorities for specific job or jobs(comma separated list).
`--users={username}`	Print the job priorities for jobs of specific user or users(comma-separated list).

To stop and remove a job from a queue, use:

```
[casper]$ scancel job-ID
```

Interactive sessions on compute nodes

Any time when it is necessary to use an interactive session on a compute node could be helpful the usage of the *srun* command.
This is the one and only allowed method to have an opened remote session on a compute node.

```
[casper]$ srun
```

Starting from the login node, use the command with the following syntax:

```
[casper]$ srun --nodes=1 --tasks-per-node=1 --pty /bin/bash
```

Command's options directly correspond with the directives of sbatch scripts (see *sbatch*).

Usage of *srun* command is strictly monitored as defined by QLOGIN ABUSE SUPPRESSION policy.

An abuse is spotted when a *srun* session is signed running 'r' but actually does not execute any calculation. This could make the schedule equality policy uncertain.

The mechanism by which the "nasty" *srun* sessions is suppressed work in this way:
every day at 3:00 AM for any running *srun* it is calculated a certain value (ratio) as the average CPU usage divided by the number of requested CPU at submission time; if this value is below a certain threshold the job is terminated by mean of *scancel*.

Graphical Sessions and X11 Forwarding

By using *srun* command it is possible to create graphical interactive sessions on the computational nodes. To enable this feature it is required to initialize *X11 Forwarding* on login command as following:

```
[Casper]$ ssh -XC username@loginNode
```

then use the command below to obtain a new session on a computational node and finally call the program intended to be used.

```
[Casper] $ srun --nodes=1 --tasks-per-node=1 --x11 --pty /bin/bash
```

The complete reference manual is available here: https://slurm.schedmd.com/

MPI job distribution

SLURM offer a large variety of option for configure how the resource should be reserved for the job. In the table below a summary of the main relevants one:

`--nodes`	Each server is referred as a node and is associated with the `--nodes` directive
`--ntasks`	Task corresponds with MPI ranks and is associated with the `--ntasks` directive
`--ntasks-per-node`	Give the possibility to control the number of task on one single node
`--cpus-per-task`	Set how many cpu per task will be recerved, when used with OMP represente the number of core for single OMP process

MPI sbatch script example

MPI case 1 [mpi1.sbatch]

```
#!/bin/bash
#SBATCH --ntasks=16
#SBATCH --cpus-per-task=1
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
```

This example would request 16 tasks, corresponding to 16 MPI ranks. These will each have 1 core. All 16 tasks will be constrained to be on a single compute node.

MPI case 2 [mpi2.sbatch]

```
#!/bin/bash
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=1
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=16
```

In this second example, 32 tasks are requested, and these are split between 2 nodes, 16 tasks on each.

MPI case 3 [mpi3.sbatch]

```
#!/bin/bash
#SBATCH --ntasks=32
```

In this third example, 32 tasks are requested and the scheduler decide how to distribute them.

MPI and OpenMP usecase [mpiOpenMP.sbatch]

```
#!/bin/bash
#SBATCH --ntasks=16
#SBATCH --cpus-per-task=4
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=4
```

In this case, we are running an application that uses multiple core for each MPI task or rank. We specify 16 tasks and 4 cores per task, for a total of 64 cores. The 16 tasks are split across 4 nodes, with the `--ntasks-per-node` ensuring that they are distributed evenly across the nodes.

Environment Module System

The software *modules* (*TACC Lmod*) enables dynamic modification of the environment variables during a user session.
The use of this software is strongly recommended in the *sbatch script* and *srun session* because it provides an easy way to use different versions of the same application; it allows user to export environment variables.

Common module command:

module list	List loaded modules
module avail	List available modules
module load {module-name}	Load the module
module unload {module-name}	Unload the module
module purge	Remove all loaded module

Example:

```
$ module avail

-------------------- /opt/ohpc/pub/moduledeps/gnu-mvapich2 ---------------------
   adios/1.11.0    mpiP/3.4.1              petsc/3.7.5        scorep/3.0
   boost/1.63.0    mumps/5.0.2             phdf5/1.8.17       sionlib/1.7.0
   fftw/3.3.4      netcdf-cxx/4.3.0        scalapack/2.0.2    superlu_dist/4.2
   hypre/2.11.1    netcdf-fortran/4.4.4    scalasca/2.3.1     tau/2.26.1
   imb/4.1         netcdf/4.4.1.1          scipy/0.19.0       trilinos/12.10.1

------------------------ /opt/ohpc/pub/moduledeps/gnu -------------------------
   gsl/2.2.1      mpich/3.2           ocr/1.0.1          pdtoolkit/3.23
   hdf5/1.8.17    mvapich2/2.2 (L)    openblas/0.2.19    superlu/5.2.1
   metis/5.1.0    numpy/1.11.1        openmpi/1.10.6

------------------------- /opt/ohpc/pub/modulefiles --------------------------
   autotools        (L)    gnu/5.4.0  (L)    pmix/1.2.3
   clustershell/1.8        ohpc       (L)    prun/1.2        (L)
   cmake/3.9.2             papi/5.5.1        singularity/2.4

----------------------- /share/apps/casper-modulefiles -----------------------
   blender/2.79                       intel/python/2.7/2017.3.053
   converge/2.3.22                    intel/python/3.5/2017.3.052
   fire/2014.2                        lammps/16Feb16
   intel/libraries/daal/2017.4.239    matlab/2017b
   intel/libraries/ipp/2017.3.196     quantum-espresso/5.4.0
   intel/libraries/mkl/2017.4.239     quantum-espresso/6.2.1      (D)
   intel/libraries/mpi/2017.4.239     starccm+/12.06.011
   intel/libraries/tbb/2017.8.239

  Where:
   L:  Module is loaded
   D:  Default Module
```

For example, to load the modules required to execute *matlab*, you can use:

```
$ module load matlab/2017b
```

To be effective, in the most of the cases, the modules' commands should be directly added in the sbatch script.

Ganglia Monitoring System

Ganglia is a scalable, distributed monitoring tool for high-performance computing systems, clusters and networks. The software is used to view either live or recorded statistics covering metrics such as CPU load averages or network utilization for many nodes.

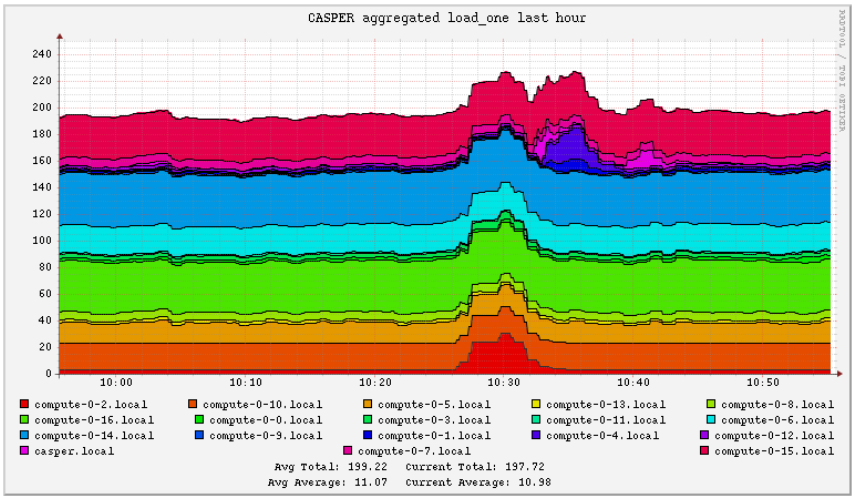

Figure 3: Ganglia - Ganglia example

Basically Ganglia is a web interface that provides information about the cluster's activities. It is available at following link:

[Casper] *http://casper.polito.it/ganglia/*

[Hactar] *http://hactar.polito.it/ganglia/*

APPENDIX A: sbatch script example

It follows a list of script templates for some of the applications installed on CASPER.

Matlab [matlab.sbatch]

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@polito.it
#SBATCH --partition=global
#SBATCH --time=hh:mm:ss
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --output=job_name_%j.log
#SBATCH --mem-per-cpu=1024M

module load matlab/2017b

matlab -r test_matlab
```

Blender [blender.sbatch]

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@polito.it
#SBATCH --partition=global
#SBATCH --time=hh:mm:ss
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --output=job_name_%j.log
#SBATCH --mem-per-cpu=1024M

module load blender/2.79

blender -noaudio -b BMW1M.blend -o BMW1M_out -F PNG -f 1 -t${SLURM_NTASKS}
```

converge [converge.sbatch]

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@polito.it
#SBATCH --partition=global
#SBATCH --time=24:00:00
#SBATCH --ntasks=96
#SBATCH --output=job_name_%j.log
#SBATCH --mem-per-cpu=2048M

module purge
module load converge/2.3.22

echo Starting on $SLURM_JOB_NODELIST with $SLURM_NTASKS processors

mpirun --bind-to core converge-2.3.22-openmpi-linux-64 $inFile > output
```

quantum-espresso [qe.sbatch]

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@polito.it
#SBATCH --partition=global
#SBATCH --time=24:00:00
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=32
#SBATCH --exclude=compute-0-8
#SBATCH --output=job_name_%j.log
#SBATCH --mem-per-cpu=2048M

module purge

module load quantum-espresso/6.2.1

CASE_IN="ausurf.in"

echo Starting on $SLURM_JOB_NODELIST with $SLURM_NTASKS processors

prun pw.x -ntg 2 -ndiag 16 -input $CASE_IN
```

APPENDIX B: generate submission scripts from a csv file

For advance tasks which requires multiple execution of the same program with different parameters, we suggest you to use following script which is able to generate multiple sbatch script file by starting from a text file containing all the parameters.

[generate.sh]

```
#!/bin/bash
#
list=$(cat ./parameters.csv)
#
for i in $list
    do
    parameter1=$(echo $i|cut -f 1 -d ',')
    parameter2=$(echo $i|cut -f 2 -d ',')
    cat << EOF > ./test-$parameter1-$parameter2.sbatch
    #!/bin/bash
    #SBATCH --job-name=job_name
    #SBATCH --mail-type=ALL
    #SBATCH --mail-user=nome.cognome@polito.it
    #SBATCH --partition=global
    #SBATCH --time=24:00:00
    #SBATCH --nodes=3
    #SBATCH --ntasks-per-node=32
    #SBATCH --output=job_name_%j.log
    #SBATCH --mem-per-cpu=2048M
    example.bin  $parameter1 $parameter2
    EOF
done
```

[parameters.csv]

```
A,1
A,2
B,1
B,2
```