On the Usage of GPU

Juan Maroñas & Manuel Villarreal

December 2020

Contents

1 Machines: General Instructions

VERY IMPORTANT: We have make a very important change in how the cluster works. Please finish reading this section for details.

Quick guide for the use of prhlt machines. Any questions / suggestions / problems contact the administrators:

- Manuel Villarreal Ruiz: maquinas@prhlt.upv.es

The idea of this management is based on having updated the drivers to the latest possible version and several possibilities in the use of cuda and cudnn. It will be giving backward compatibility as much as possible. It will be sought that the machines always have the same version of everything (kernel, Linux libraries ...). In addition, several versions of each software are provided.

Important: For the transfer of machines first contact the administrators before proceeding to turn off, to know if someone is using them at that time. For any other problem contact us.

Updates and more: Any management on the machines (except very specific exceptions or necessity[1]) will be made on Fridays. This exclude solving problems to guarantee the correct usage of the machines, as example reinstalling drivers or rebooting.

New accounts: When requesting a new account the default password is 1234. You can change it executing passwd on the terminal.

How should I launch in the machines?: Please read section 2 for details. Read section 3 for tips on how to properly set your enviroments or how to use the software installed. Go over section 6 for deprecated installation. Y

Recent News :

- **Next Release:** .Ubuntu 20 is installed.
- **Software:** All the software from /usr/local has been deleted. Check the Github account here for examples on how to install tools. I wont give support to all this software anymore, hence I encourage the users of the different tools to update this files accordingly.
- **CUDA:** Only cuda 10.1 is available for ubuntu 20 at the moment. It is installed by the ubuntu manager which means it is installed in /usr/bin and /usr/lib. This means there is no need to set LD_LIBRARY_PATH. When nvidia releases CUDA for ubuntu20 I will install things as before

[1] For necessity we mean an important deadline as example

- **CUDNN:** Cudnn is installed in /usr/local as always, but only the versions for CUDA10>. You need LD_LIBRARY_PATH for this.
- **Comming soon:** Cuda11
- **Python:** Python2.7 is no longer supported. Virtual-env is no longer supported, only conda. I have installed python3.x for x in 5,6,7,8,9.
- **Deprecated section:** This section has been removed from this file.

Access NAS: If you want additional space or to centralized the data there are two NAS available. Contact Joan Andreu Sanchez.

Please keep in mind:

- It is important to keep in mind that the machines are used by several people and that the memory is not unlimited. It is appreciated to release the memory from time to time with those models that we no longer use.

- On the other hand, bear in mind that there are software (such as pytorch) that when you install them download everything you need (that is, they also download cuda) so the software can be easily duplicated in the machines taking up a lot of memory. For example, if each user installs locally kaldi then it will occupy 14GB uselessly. Keep in mind this to save memory.

2 Correct usage of machines

This section provides a set of instructions on how the machines should be used to guarantee a good usage and share between the members.

- Before launching you should check with two commands if the machine is available or not:
  - htop → would let you know if the machine is running any process.
  - nvidia-smi → would let you know if someone is launching on a GPU.
- **Why is not recommended to launch if someone is launching?:**
  - Let's suppose that a GPU can hold 8GB of memory and a process is taking 5GB. Imagine your process takes only 2GB. This means that you could launch your process and the GPU could run both of them (because we have 8GB and only 5GB are occupied). Now imagine that you have launched but the initial process requires more memory (let say 2 additional GB, 7GB in total) and you have launched your 2GB memory process, then the process from the other user will die because there is only 1GB left. This will affect your team mate.

  - Although the bottleneck to launch in a GPU is the amount of memory available, you can severely down-speed the processes because all of them will be sharing the same CUDA Cores. Is the same as if you have a 120GB Ram computer but you have a single CPU. Launching many processes at the same time will affect the whole performance.
  - Very important: Launching several processes can require additional power from the power supply. Some of the machines do not have enouch power supply (1000Wats when 1200 is required) and this means that if the computer request more power than what the power supply can provide, then the machine will reboot. I asked this in a forum one year ago: https://discuss.pytorch.org/t/nn-dataparallel-crashes/46023 . Keep in mind that you can launch in a GPU if it is free, but it is not recommended to launch two processes in the same GPU to avoid this power supply problem.

- In the group documentation you have the emails from all the GPU users. This means that you can easily coordinate with those that have an account on the same GPUs you have.

3 Installed libraries

Before proceeding, I recommend going over section 4 (Appendix 1) for an insight on compilation and linking. You need to know how linux work in order to get many software properly working. As a general rule, first look for the library in the software that we want to use and if it is not then check if it is a Unix library that needs to be installed. You can consider the following steps:

- If the library is not found and it is a generic Linux library (I have previously searched the internet for that), the administrators are contacted to install it.
- If the library is not found but it is already installed, we can find its directory by executing grep on the path to the software we want to use, and then add the route to the library to our environment variable. Check section 4 if you do not have idea of what I am talking about.

3.1 NVIDIA LIBRARIES

The machines will have the Nvidia CUDA library and the cuDNN libraries for acceleration in Deep Learning. The routes to the directories of these libraries are in:

- CUDA X: "/usr/local/cuda-X/"
- cuDNN : "/usr/local/cudnn/cuda-X"

Where cuda-X represents the desired cuda version (example cuda-8). Each cudnn library (specific for each cuda) is installed under the cuDNN directory. For example to use cudnn7 for cuda 8 you must go to /usr/local/cudnn/cuda8/ and to use it for cuda9 you must go to /usr/local/cudnn/cuda9/

The following table shows the versions of CUDA and cuDNN available **Important note. Since ubuntu20 only cuda11.1 is available. I keep cudnn for older cuda10 versions but that will be probably removed soon. If there is a need to have cuda10 in the machines let me know and will reinstall through apt-get, although I prefer to keep cuda installed only through the instalers provided by Nvidia**:

cuda-10.0	**cuda-10.1**	**cuda-10.2**	**cuda-11.1**
7.3.0	-	-	-
7.3.1	-	-	-
7.4.1	-	-	-
7.6.5	7.6.5	7.6.5	-
-	-	-	8.0.4

Go inside the cuDNN directories of the cuda version you want to use to check for the name of that version. For example to use cudnn7 version 3.1 in cuda 10 you run:

- ls /usr/local/cudnn/cuda10/

And the output will show: dnn7.3.0 dnn7.3.1 dnn7.4.1 dnn7.6.5

Obviously dnn7.3.1 is your choice, so finally your desired library is under:

- /usr/local/cudnn/cuda10/dnn7.3.1/

Within these folders we find the directory with the binaries and the directories with the libraries. Depending on how you compile your software we may require some steps or others. And depending how your software (for example Tensorflow) is made. We provide a further review in section 4 (Appendix 1) on how UNIX manage compilation and linking. In general and for cuda 10 (note that it is the same for the rest):

- binaries (compiler...): "/usr/local/cuda-10.0/bin/"
- libraries: "/usr/local/cuda-10.0/lib64/"
- headers: "/usr/local/cuda-10.0/include/"

In the case of cudnn:

- libraries: "/usr/local/cudnn/cuda10/dnn7.3.1/lib64/"
- headers: "/usr/local/cudnn/cuda10/dnn7.3.1/include/"

In libraries we find dynamic and static versions of them.

3.2 Software and Unix libraries

See Github README file for details.

3.3 Nvidia Docker

We have installed nvidia-docker. If you want to use it contact the administrator to be added to the docker group.

3.4 Conda and not pip

Pip is not supported anymore, just conda.

Conda is installed in /usr/local/anaconda3/ . From now own we will refer to this path as conda_path.

There are several ways to create condas enviroments. Depending on how you do it it will install it to .local, to the root directory of conda.

To run conda you must type /usr/local/anaconda3/bin/conda.

If you want to add conda to your .bashrc so that conda is run automatically when you type 'conda' then you have to run

```
/usr/local/anaconda3/bin/conda init
```

With this you can execute any of the commands below withouth having to use the absolute path /usr/local/anaconda3/bin and just use conda.

To create a python enviroment using conda type the following:

/usr/local/anaconda3/bin/conda create -y –no-default-packages –prefix my_enviroment python=3.7

This will create a conda enviroment using python 3.7 named my_enviroment.

To activate the enviroment type:

source /usr/local/anaconda3/bin/activate my_enviroment

Then you can install whatever package you want:

pip install tensorflow_gpu

To deactivate it type:

source /usr/local/anaconda3/bin/deactivate

NOTE: Examples on how to use conda to install software such as PyTorch or Tensorflow can be found in this repository.

NOTE: You might need to configure correctly the enviroment variables so that CUDA CUDNN or other libraries are found. See the appendix.

3.5 Merging Python Virtual Enviroments

Some tips to merge and use two different python enviroments. There are two solutions, one unix based and one python based.

This is a python based solution. We create in our home, lets suppose /home/prhlt/, an enviroment variable (it could be created with conda or just installed directly in .local folder):

- cd /home/prhlt/
- virtualenv -p python2.7 myenviroment
- source myenviroment/bin/activate
- pip install scipy
- deactivate

We create an enviroment with scipy installed. Now, to use both environments it is as easy, just follow the next steps. We activate our TF enviroment (follow TF steps):

- source /usr/local/anaconda3/bin/activate /usr/local/tensorflow/python2.7/1.4.1

Then, in your python script:

```
#My new state of the art model
import tensorflow
import scipy #if you do it here it will raise error
import sys
sys.path.extend(["/home/prhlt/myenviroment/lib/python2.7/site-packages/"
,"/usr/local/keras/python2.7/lastversion/lib/python2.7/site-packages/"])

import scipy #after adding the myenviroment lib path everything works perfectly
import keras
import tensorflow
```

A unix based solution: There is a unix based solution (thanks to Mario for finding it). Suppose we create an enviroment to install scipy or whatever:

- export LD_LIBRARY_PATH=LD_LIBRARY_PATH:/usr/local/cuda-9.0/lib64/
- export LD_LIBRARY_PATH=/usr/local/cudnn/cuda8/dnn6/lib64:$LD_LIBRARY_PATH # do this if necessary
- virtualenv -p /usr/bin/python3.5 myapp
- you activate virtualenv and install whatever you need (pip install scipy)
- deactivate

Now after you have it just type:

- realpath /usr/local/tensorflow/python3.5/1.4.1/lib/python3.5/site-packages > myapp/lib/python3.5/site-packages/base_venv.pth
- realpath /usr/local/keras/python3.5/lastversion/lib/python3.5/site-packages >> myapp/lib/python3.5/site-packages/base_venv.pth

And now you can use keras and your enviroment with tensorflow without having to modify your .py files. **Just activate** you enviroment myapp and import keras and tensorflow.

This can also be done with conda enviroments, or with any path in your system. For instance with conda enviroments we can do:

```
$conda_path/bin/create --name /tmp/env_name
realpath /usr/local/tensorflow/python3.5/1.4.1/lib/python3.5/site-packages >
/tmp/env_name/lib/$python_version/site-packages/base_env.pth
source $conda_path/bin/activate env_name
```

3.6 Launch as fast as Joan Andreu

If you want to launch in several machines at the same time from your computer there is an easy way to do it. The trick is to use dsh command. I put here the tutorial provided by Lorenzo:
First install dsh

```
sudo apt-get install dsh
```

The configure it (for secure connection):

```
sudo sed -i "s:remoteshell=rsh:remoteshell=ssh:g" /etc/dsh/dsh.conf
```

Create the machines.lst file. There are two options, first we can add the machines to the default file /etc/dsh/machines.list or create a new one and use the option -f to redirect dsh to it. The file is very simple: a txt file where each line is a machine in the format username@machine address as in a normal ssh. Example:

```
lquirosd@150.222.222.44
lquirosd@150.222.222.44
lquirosd@150.222.222.45
```

Password less login. Because we are using ssh a password is required for each time you login to a machine. To prevent that we can enable password-less ssh communication from the host where you've installed dsh:

1. Run ssh-keygen -t rsa on the host machine.
2. this command will generate a id rsa.pub file. Copy the contents of this file to the $USER/.ssh/authorized keys file on each of the machines listed on the machines.lst file.
3. run ssh to each machine to ensute it works as expected.

Execute commands in all the machines. To execute any command in the machines, just use the option -c in the command. For example, to check the memory usage in some GPU we can use the command

```
nvidia-smi --query-gpu=memory.total,memory.used,memory.free --format=csv
```

, to run it on all the machines:

```
dsh -f <pointer to machines file> -M -c "nvidia-smi
--query-gpu=memory.total,memory.used,memory.free
--format=csv"
```

4 Appendix 1. An Insight on Linux compilation and link stages

There are two basic environment variables for the use of machines: PATH y LD_LIBRARY_PATH.

As we know, the systems based on linux have certain directories where the executables of the system are placed by default, and therefore, is where the operating system looks for said binaries. For example we have: /bin, /sbin, /usr/local/bin, /usr/local/sbin... When we want the operating system to look for binaries in other folders we must add it to the path variable. For example, if I want the Nvidia compiler to be detected automatically, We can do:

export PATH=$PATH:/usr/local/cuda-8.0/bin/

So when we run a program and it gives us the typical error: "Can not found NVCC" is because it is not found in $PATH.

We also know that software is usually supplied by libraries and that for several reasons dynamic linkage is advantageous. So the dynamic linking phase has two

stages: one is to find the libraries while compiling and another is to find the libraries when it is executed (this is done by the linker /usr/bin/ld).

Again the linux based systems have default directories where the linker finds the libraries that we install: /lib, /usr/lib, /lib64. In the compilation phase there are two ways to tell the linker where to find possible libraries.

One is through the flag -L and another is through the flag -rpath. The difference is very simple: -L indicates only where the libraries are but it does not save the directory where they are in the binary, while -rpath saves it. Therefore to a software compiled with -L we must indicate when we execute (in the runtime phase) where the libraries are and to a software compiled with -rpath no, since having saved the directory the linker automatically goes there to search.

For that reason, the configuration of our enviroments depends on how our compiled software or our precompiled software (TF, torch...) is done. We use variable LD_LIBRARY_PATH to indicate the system where to search for other libraries when they are not found in the default directories. For example, for those softwares that make use of cudnn we must place here where the specific version of cudnn is. Even if our software is already compiled (like tensorflow) we must tell the linker where to find the libraries. Note that compiling with rpath does not make sense when distributing software because we may place cudnn wherever we want. As example place (to use cudnn6 for cuda8):

export LD_LIBRARY_PATH=/usr/local/cudnn/cuda8/dnn6/lib64/

Many of the linking stage can be saved by using ldfconfig. Example:

sudo echo "/path_to_library" >> /etc/ld.so.conf; sudo ldconfig

5 Acknowledgment

Thanks to Mario Parreño for testing software and to Lorenzo Quiros and Mario for helpfull discussion and recommendations. Also thanks to Miguel and Jose for support in many aspects.

6 Account Distribution

Our idea for managing accounts is based on people that works in the same area are place in the same machine. With this we try to avoid duplicating big datasets and encourage people to organize themselves when launching in the same machine.

6.1 IP and Hardware Address

From gpu15 to gpu26, these machines have rtx-2080 and thus only cuda10 will run.

Machine	IP Adress	Number of GPUs	HWaddress	BIOS MODE	wake up
gpu1	user@prhltgpu.dsic.upv.es	1	d0:50:99:a8:59:46	BIOS	no
gpu2	user@prhltgpu2.dsic.upv.es	1	d0:50:99:ab:7b:9e	BIOS	
gpu3	user@prhltgpu3.dsic.upv.es	1	38:d5:47:12:72:2e	BIOS	
gpu4	user@prhltgpu4.dsic.upv.es	1	38:d5:47:12:74:09	BIOS	
gpu5	user@prhltgpu5.dsic.upv.es	1	38:d5:47:12:65:7c	BIOS	
gpu6	user@prhltgpu6.dsic.upv.es	1	38:d5:47:12:70:72	BIOS	
gpu7	user@prhltgpu7.dsic.upv.es	1	38:d5:47:12:68:37	BIOS	
gpu8	user@prhltgpu8.dsic.upv.es	1	70:8b:cd:ab:fe:22	BIOS	
gpu9	user@prhltgpu9.dsic.upv.es	1	70:8b:cd:ab:fe:24	BIOS	
gpu10	user@prhltgpu10.dsic.upv.es	1	70:8b:cd:ac:06:c2	BIOS	
gpu11	user@prhltgpu11.dsic.upv.es	2	e0:d5:5e:8d:3e:57	UEFI	yes
gpu12	user@prhltgpu12.dsic.upv.es	2	e0:d5:5e:8d:3e:86	UEFI	yes
gpu13	user@prhltgpu13.dsic.upv.es	2	e0:d5:5e:8b:9e:55	UEFI	yes
gpu14	user@prhltgpu14.dsic.upv.es	2	e0:d5:5e:8d:3e:94	UEFI	yes
gpu15	user@prhltgpu15.prhlt.upv.es	2	30:9c:23:61:77:9d		
gpu16	user@prhltgpu16.prhlt.upv.es	2	30:9c:23:61:7a:79		
gpu17	user@prhltgpu17.prhlt.upv.es	2	30:9c:23:62:60:e9		
gpu18	user@prhltgpu18.prhlt.upv.es	2	30:9c:23:61:78:32		
gpu19	user@prhltgpu19.prhlt.upv.es	2	30:9c:23:61:76:d4		
gpu20	user@prhltgpu20.prhlt.upv.es	2	30:9c:23:61:77:8a		
gpu21	user@prhltgpu21.prhlt.upv.es	2	30:9c:23:61:78:6a		
gpu22	user@prhltgpu22.prhlt.upv.es	2	30:9c:23:61:7a:28		
gpu23	user@prhltgpu23.prhlt.upv.es	2	30:9c:23:61:77:fb		
gpu24	user@prhltgpu24.prhlt.upv.es	2	30:9c:23:61:76:fe		
gpu25	user@prhltgpu25.dsic.upv.es	2	04:d4:c4:57:2d:47	UEFI	
gpu26	user@prhltgpu26.dsic.upv.es	2	04:d4:c4:57:2c:a8	UEFI	

6.2 Accounts deleted but preserves /home

Some accounts are deleted. However, we let the /home directories in the corresponding machines, to recover necessary datasets, experiments... When deleting and account this must be requested, otherwise the /home folder would also be deleted.

6.3 Inactive machines

Inactive machines due to several problems:

- GPU14: inactive due to unknown reasons
- GPU21: inactive due to operative system problems

6.4 Closed machines

6.5 Number of GPUS per user

Responsable	User/username	1GPU Computer	2GPUs Computer	Total GPUs	Total Computers
Carlos Martinez	David Gimeno/dgimeno (dagigo1@inf.upv.es)	-	4	8	4
Paolo	Gretel/gretelliz (gretelliz92@gmail.com)	-	4	8	4
Paolo	Reynier Ortega / rortega (reynier.ortega@gmail.com)	1	3	7	4
Paolo	Angel De Paula / adepau (adepau@doctor.upv.es)	1	5	11	6
Paolo	Ipek Baris / ibar (ibarsch@doctor.upv.es)	-	2	4	2
Paolo	Ivan Grubisic / igrubis (igrubis@upvnet.upv.es)	-	2	4	2
Paolo	Ana-Maria Bucur / banamar (banamar@doctor.upv.es)	-	3	5	3
Paolo	Meryem / meryem (meryem.AIT-MOUT@etu.univ-amu.fr)	1	1	3	2
Paolo	Yaqine / yaqine (saad.yaqine@etu.univ-amu.fr)	1	1	3	2
Paolo	Giulia Rizzi / grizzi (grizzi1@doctor.upv.es)	0	2	4	2
Jon	Jon Ander/jon jon@dsic.upv.es	2	3	8	5
Jon	Daniel Chorro Juan/dchorro (dchojua@inf.upv.es)	2	-	2	2
Jon	Jose Javier Calvo / jojaca (jocalmo@inf.upv.es)	1	-	1	1
Jon	Francisco Tomás García / ftgarrui (ftgarrui@inf.upv.es)	2	-	2	2
Roberto	Mario parreño/maparla (maparla@prhlt.upv.es)	-	6	12	6
Roberto	Salva Carrion/scarrion (salcarpo@inf.upv.es)	-	3	6	3
Roberto	Daniel Parres/dparres (dparres@prhlt.upv.es)	-	5	10	5
Roberto	José Arias/joarmon1 (joarmon1@inf.upv.es)	-	2	4	2
Roberto	Jorge Juan/jjuagon (jjuagon@posgrado.upv.es)	2	1	4	3
Roberto	Yuri Vladimir Huallpa/yhuavar (jjuagon@posgrado.upv.es)	2	0	2	2
Roberto	Antonio Blasco Calafat/anblaca (anblaca@posgrado.upv.es)	0	2	4	2
Enrique	Vicente Bosch/vbosch (vbosch@gmail.com)	4	-	4	4
Enrique	Lorenzo Quirós/lquirosd (lorenzoqd@gmail.com)	3	5	13	8
Enrique	Jose Prieto/jprieto (joprfon@posgrado.upv.es)	1	5	11	6
Enrique	Josep Salvador / jsalvador (josep100998@gmail.com)	1	1	3	2
Vero	Vero/vromero (vromero@prhlt.upv.es)	1	-	1	1
Moises	Moises/mpastorg (mpastorg@upvnet.upv.es)	1	2	5	3
Emilio	Emilio Granell / egranell (egranell@upv.es)	2	-	2	2
José Andrés	José Andrés / joanmo2 (joanmo2@prhlt.upv.es)	2	4	10	6
Joan Andreu	Joan Andreu /jandreu (jandreu@prhlt.upv.es)	4	1	6	5
Joan Andreu	David Vilanova/dvillanova (davilap@inf.upv.es)	-	1	2	1
Joan Andreu	Dan Atei / danitei (daan4@inf.upv.es)	2	1	4	3
Joan Andreu	Manuel Villarreal / mavilrui (mavilrui@inf.upv.es)	1	7	15	8
Joan Andreu	Yaiza Miñana / yaimiama (yaimiama@inf.upv.es)	2	-	2	2
Paco Casacuberta	P Casacuberta / fcuberta (fcn@prhlt.upv.es)	2	1	4	2
Paco Casacuberta	Miguel Domingo/midobal (midobal@prhlt.upv.es)	3	3	9	6
Paco Casacuberta	Angel Navarro / annamar (annamar8@posgrado.upv.es)	-	4	8	4
Paco Casacuberta	Angel Andujar / andujar (anancar1@inf.upv.es)	1	1	3	2
Miguel Domingo	Sergio/sgomgon (sgomgon@inf.upv.es)	0	2	4	2
Paco Casacuberta	Nora Hafidi / nohalo (nohalo@inf.upv.es)	2	-	2	2
Paco Casacuberta	Adrian Vazquez / avaba (adrianvazquezbarrera@icloud.com)	2	-	2	2
Alejandro Hector Toselli	Alejandro Hector Toselli / ahector (ahector@prhlt.upv.es)	-	4	8	4
Damic Korencic	Damic Korencic / dkorenc (dkorenc@upvnet.upv.es)	-	2	4	2
Mariona Coll Ardanuy	Mariona Coll Ardanuy / mcoll (mcoll@prhlt.upv.es)	-	2	3	2
Benedetta Togni	Benedetta Togni / btogni (btogni1@upvnet.upv.es)	-	2	4	2

GPU1: fcuberta - lquirosd - midobal - vbosch

GPU2: jon - jprieto - lquirosd - yaimiama

GPU3: danitei - egranell - jandreu - lquirosd - mpastorg - vbosch - jjuagon

GPU4: jandreu - vbosch - nohalo - yaimiama - jjuagon

GPU 5: dchorro - midobal - vbosch - danitei - nohalo - avaba

GPU 6: egranell - vromero - adepau - joanmo2

GPU 7: fcuberta - jjsjunquera - jon - midobal - jandreu - martnbo

GPU 8: andujar - dchorro - fenuel - jojaca - rortega

GPU 9: joanmo2 - ftgarrui - yaqine

GPU10: jandreu - ftgarrui - yhuavar - meryem - mcoll

GPU11: fcuberta - gretelliz - jsalvador - yhuavar - meryem

GPU12: jon - jprieto - lquirosd - maparla - joanmo2

GPU13: annamar - rortega - jprieto - dparres - btogni

GPU14: annamar - jprieto - dparres - yaqine - btogni

GPU15: mavilrui - dparres - jprieto - ahector - anblaca

GPU16: mavilrui - jprieto - dgimeno - ibar - banamar

GPU17: annamar - maparla - adepau - joarmon1 - ahector - grizzi

GPU18: ibar - jon - maparla - dgimeno - banamar - grizzi

GPU19: gretelliz - lquirosd - midobal - mavilrui - jsalvador - danitei

GPU20: gretelliz - scarrion - mavilrui - lquirosd - dparres - adepau

GPU21: lquirosd - scarrion - dkorenc - midobal - mpastorg - igrubis

GPU22: dgimeno - banamar - joanmo2 - dparres - igrubis - rortega - sgomgon

GPU23: dvillanova - maparla - mavilrui - rortega - joarmon1 - adepau

GPU24: jon - adepau - ahector - rortega - jjuagon - mpastorg

GPU25: lquirosd - scarrion - dkorenc - midobal - joanmo2 - anblaca - sgomgon

GPU26: annamar - mavilrui - adepau - joanmo2

	Number of Users
GPU1	4
GPU2	5
GPU3	7
GPU4	5
GPU5	6
GPU6	4
GPU7	6
GPU8	5
GPU9	3
GPU10	5
GPU11	5
GPU12	5
GPU13	5
GPU14	5
GPU15	5
GPU16	6
GPU17	6
GPU18	6
GPU19	6
GPU20	6
GPU21	6
GPU22	6
GPU23	6
GPU24	6
GPU25	6
GPU26	4