Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Guida operativa a delle modalità d'uso del cluster IBM

Francesco Gargiulo, Gennaro Oliva

Rapporto Tecnico N: **RT-ICAR-NA-2020-03** ***Data:*** **Novembre 2020**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Guida operativa a delle modalità d'uso del cluster IBM

Francesco Gargiulo, Gennaro Oliva

Rapporto Tecnico N: **RT-ICAR-NA-2020-03** ***Data:*** **Novembre 2020**

Guida operativa a delle modalità d'uso del cluster IBM

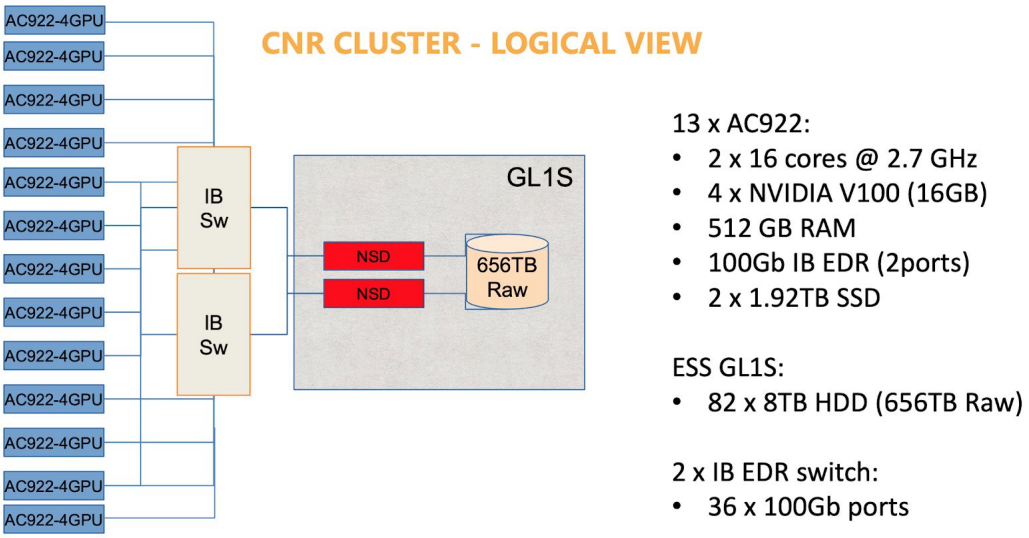
Francesco Gargiulo, Gennaro Oliva
Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche (ICAR-CNR)
Via Pietro Castellino, 111 – 80131 Napoli, Italia
{francesco.gargiulo, gennaro.oliva}@icar.cnr.it

1. Premessa

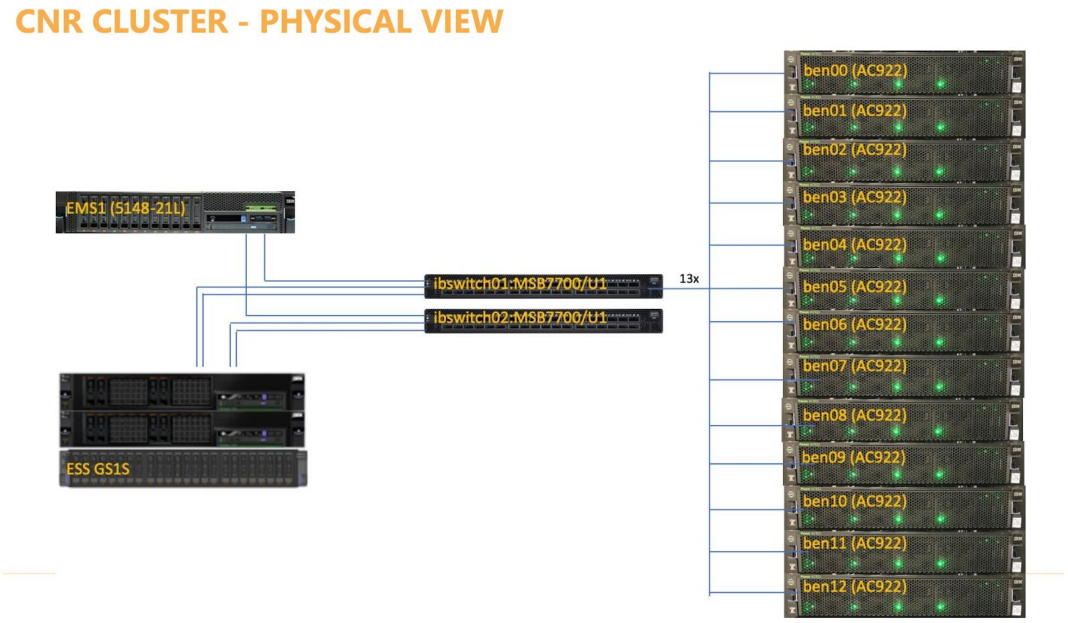
Oggigiorno è fondamentale dotarsi di infrastrutture hardware per poter fare ricerca scientifica di livello internazionale. A tal fine l'ICAR si è recentemente munito di un cluster prodotto dalla IBM che può supportare i ricercatori nelle loro sperimentazione in diversi scenari applicativi simulabili solo in ambito High Performance Computing come ad esempio: l'Intelligenza Artificiale ed in particolare le complesse reti prodotte attraverso il paradigma del Deep Learning, la gestione e l'analisi di grosse moli di dati (Big Data Analytics), lo studio della sicurezza in senso lato, ad esempio (blockchain) e molto altro ancora. In sintesi, oggi, per validare la maggior parte di queste metodologie sono necessarie ingenti risorse di calcolo, in particolar modo l'accelerazione fornita dalle GPU.
In questo documento sarà fornita al lettore una panoramica del cluster IBM di nuova installazione; l'obiettivo principale è quello di fornire degli spunti per poter rendere tutti gli utenti abilitati, rapidamente operativi nell'usare le potenzialità offerte da questa risorsa. Il documento è strutturato come segue, nella sezione 2 verrà fornita una panoramica sulle specifiche hardware del cluster ed una chiave di lettura sulle motivazioni che hanno portato la IBM a determinate scelte architetturali; nella sezione 3 si parlerà del gestore dei job presente sul cluster e delle sue modalità di utilizzo per alcuni scenari tipici; nella sezione 4 si descriverà la gestione delle repository software e dell'ambiente conda nel quale sono raccolti i pacchetti ottimizzati dalla IBM per la piattaforma hardware in questione; nella sezione 5 si descriverà una possibile modalità d'uso della piattaforma in emulazione di una workstation e, infine, nella sezione 6 verrà fornita la metodologia attraverso la quale è possibile pianificare ed eseguire un'ottimizzazione dei parametri tramite l'esecuzione di molteplici test in parallelo.

2. Panoramica delle specifiche hardware

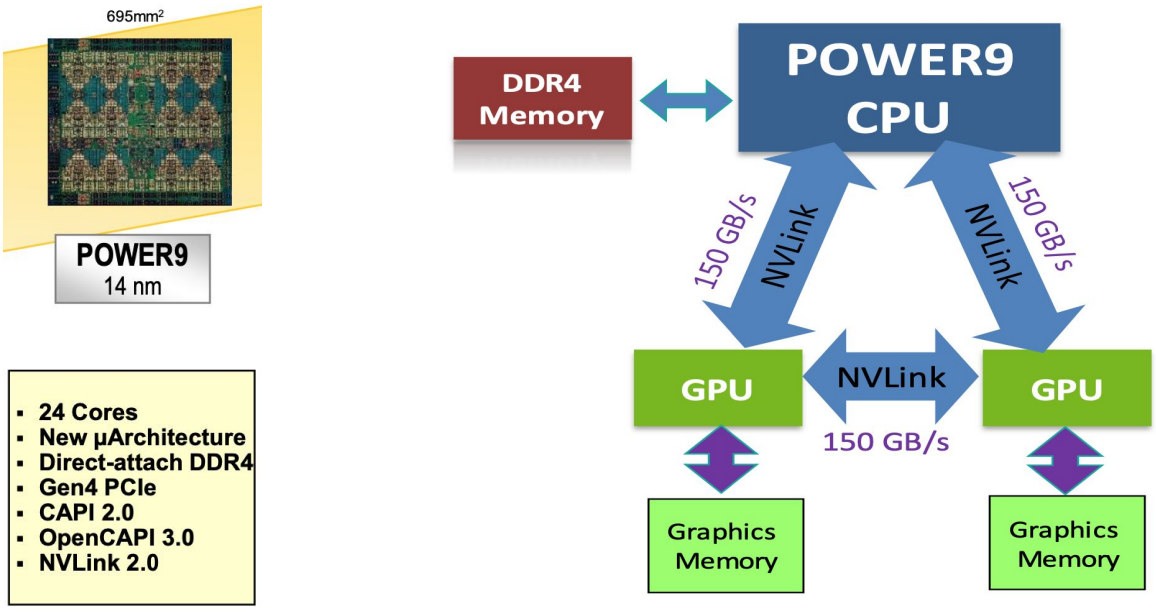
Il cluster IBM è formato da 13 nodi operazionali AC922, un'unità di storage ESS GL1S e due switch, come rappresentato nella seguente figura.



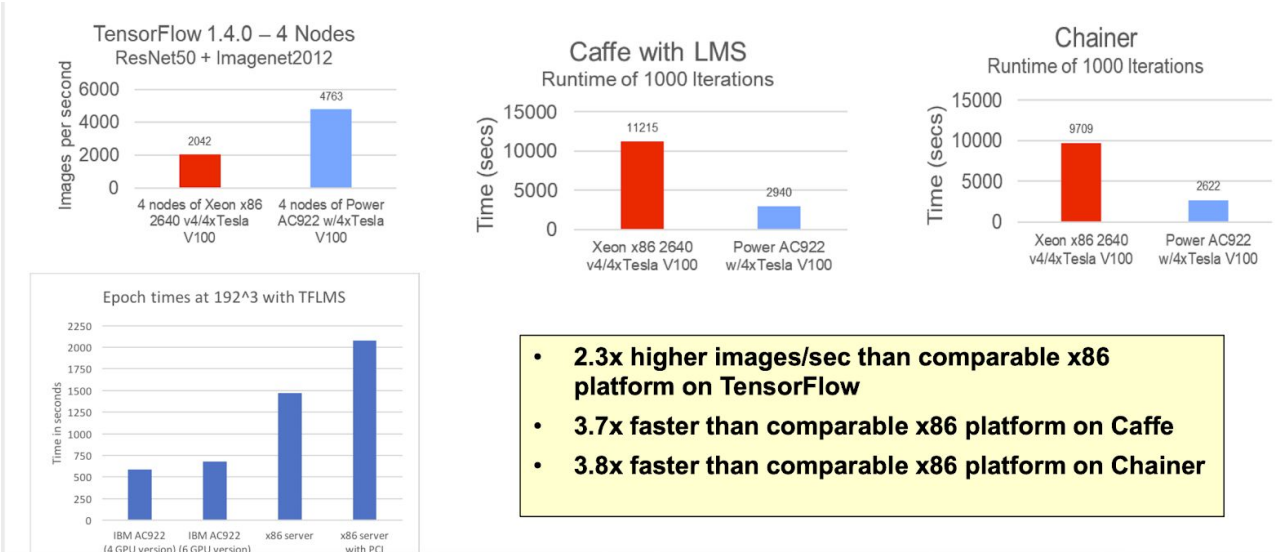
A livello fisico, i 13 nodi operazionali sono collegati allo switch01 che ha anche una doppia connessione verso lo storage (ESS GS1S) e una connessione al front-end (EMS1), lo switch02 è collegato solo al front-end e, in doppia connessione, al NAS. I nomi simbolici dei nodi di calcolo vanno da *ben00* fino a *ben12*.



Tutti i nodi operazionali sono provvisti di 4 GPU Nvidia V100 con 16 GB di RAM e due processori POWER9, questi processori hanno il vantaggio principale rispetto alle più comuni soluzioni X86 di una gestione della memoria più efficiente, specialmente nella possibilità da parte delle GPU di usare tutta la memoria di sistema per caricare i modelli, *Large Memory Support* (LMS).



Con questo tipo di configurazione vengono dichiarati dei risultati molto incoraggianti rispetto all'analoga architettura INTEL, come si può evincere dai grafici in basso prodotti dalla IBM stessa.



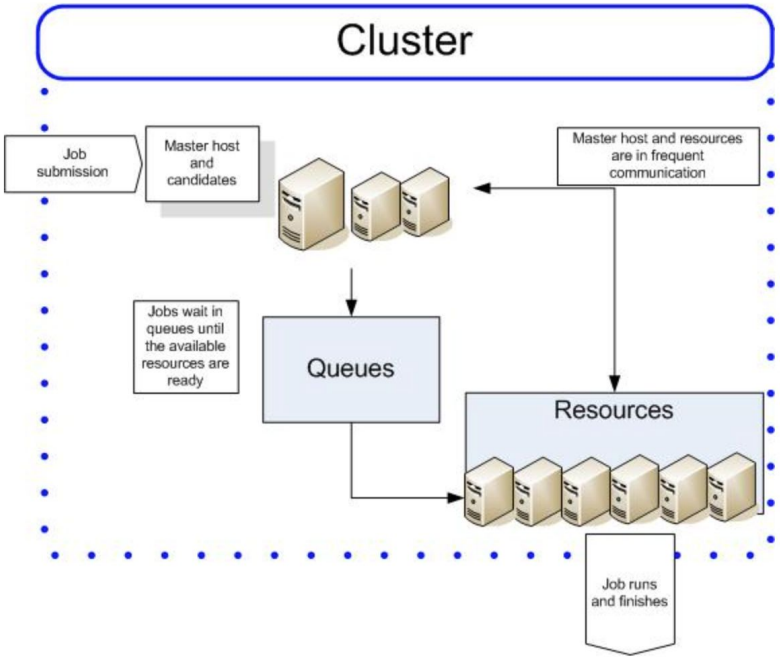
Di fatto, per ottenere queste prestazioni, bisogna usare le versioni dei tool adattate all'architettura e, nella maggioranza dei casi, non sono le più recenti disponibili.

3. La gestione dei job tramite LSF

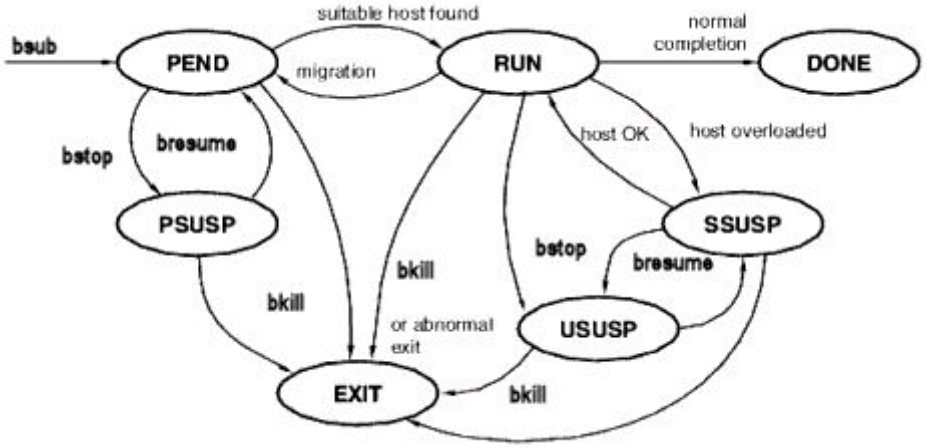
Platform Load Sharing Facility (o più semplicemente LSF) è un gestore di job, tutta la documentazione è i comandi sono consultabili dal link al sito della IBM[1]. Per gli scopi di questo documento saranno elencati i concetti e le funzionalità chiave che permetteranno un primo rapido utilizzo dello strumento. Nella figura in basso vengono illustrati in maniera grafica i concetti chiave del gestore. La sottomissione del JOB viene fatta ad un JOB_MASTER che, di solito, è un processo che gira sul MASTER_HOST che, nel nostro caso, è il frontend `ems1`. Il JOB_MASTER pone la richiesta in una delle code (QUEUES) in attesa che si liberino le risorse richieste per la corretta esecuzione del JOB. Una volta che questo avviene il JOB viene lanciato su

[1] https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_welcome/lsf_welcome.html

un sottoinsieme delle risorse del cluster che sono state richieste. Quando il JOB termina restituisce l'output. In tutte le fasi, il JOB_MASTER sta in costante comunicazione con le RISORSE monitorando lo stato di ognuna di esse e rendendo questa informazione disponibile agli utenti del cluster.



Nel diagramma in basso, vengono descritti gli stati nei quali possono trovarsi i JOB durante le fasi di gestione della richiesta, sottolineando anche la natura delle transizione tra i diversi possibili stati.



Ad esempio, al momento della scrittura di questa parte del documento, sul cluster è presente la seguente situazione:

```
[(base) [gargiulo@ems1 ~]$ bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
wmla            100  Open:Active       -    -    -    -     0     0     0     0
admin            50  Open:Active       -    -    -    -     0     0     0     0
priority         43  Open:Active       -    -    -    -     0     0     0     0
short            35  Open:Active       -    -    -    -     0     0     0     0
dataq            33  Open:Active       -    -    -    -     0     0     0     0
normal           30  Open:Active       -    -    -    -     0     0     0     0
interactive      30  Open:Active       -    -    -    -     1     0     1     0
(base) [gargiulo@ems1 ~]$
```

c'è un solo job attivo nella coda "interactive", ed è usato per emulare una workstation come vedremo nella prossima sezione.

Per avere maggiori dettagli circa questo job basta eseguire il seguente comando:

```
[(base) [gargiulo@ems1 ~]$ bjobs
 JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
 16463   gargiul RUN   interactiv ems1        ben02       /bin/bash  Nov  2 18:19
 (base) [gargiulo@ems1 ~]$
```

L'output di questo comando (`bjobs`) ci mostra che questo processo è in stato `RUNNING` sull'host `ben02` e sta eseguendo il processo`/bin/bash,` si noti che vengono visualizzati solo i job attivi per l'utente e non tutti quelli che girano sul sistema.
Per avere un'idea dell'occupazione delle risorse dell'intero cluster è possibile eseguire il seguente comando:

```
[(base) [gargiulo@ems1 ~]$ lsload
 HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
 ben06               ok   0.0   0.0   0.0   0%   0.0   0  3087 1660G  3.9G  464G
 ben00               ok   0.0   0.0   0.0   0%   0.0   0  3118 1654G  3.9G  460G
 ben03               ok   0.0   0.1   0.0   0%   0.0   0  3094 1660G  3.9G  460G
 ben07               ok   0.0   0.1   0.0   0%   0.0   0  3092 1660G  3.9G  464G
 ben05               ok   0.0   0.0   0.0   0%   0.0   0  3093 1660G  3.9G  464G
 ben11               ok   0.0   0.1   0.0   0%   0.0   0  3086 1660G  3.9G  528G
 ben02               ok   0.3   0.2   0.3   0%   0.0   0  1542 1660G  3.9G  462G
 ben09               ok   0.7   0.0   0.0   0%   0.0   0  3092 1660G  3.9G  460G
 ems1                ok   0.7   0.7   0.7   3%   0.0   1     0   97G  7.8G 35.1G
 ben08           unavail
 ben04           unavail
 ben01           unavail
 ben10           unavail
 (base) [gargiulo@ems1 ~]$
```

Oppure, se si è interessati all'occupazione delle gpu, si può optare per la variante:

```
(base) [gargiulo@ems1 ~]$ lsload -gpuload
HOST_NAME       gpuid   gpu_model   gpu_mode  gpu_temp   gpu_ecc  gpu_ut  gpu_mut gpu_mtotal gpu_mused   gpu_pstate   gpu_status   gpu_error
ben00               0 TeslaV100_S        0.0       35C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       37C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       39C        0.0      0%       0%      15.7G        0M            0           ok           -
ben06               0 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       39C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       36C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       41C        0.0      0%       0%      15.7G        0M            0           ok           -
ben03               0 TeslaV100_S        0.0       33C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       31C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       33C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       35C        0.0      0%       0%      15.7G        0M            0           ok           -
ben05               0 TeslaV100_S        0.0       40C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       42C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       41C        0.0      0%       0%      15.7G        0M            0           ok           -
ben07               0 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       42C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       42C        0.0      0%       0%      15.7G        0M            0           ok           -
ben11               0 TeslaV100_S        0.0       39C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       42C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       44C        0.0      0%       0%      15.7G        0M            0           ok           -
ben02               0 TeslaV100_S        0.0       36C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       37C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       35C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       40C        0.0      0%       0%      15.7G        0M            0           ok           -
ben09               0 TeslaV100_S        0.0       38C        0.0      0%       0%      15.7G        0M            0           ok           -
                    1 TeslaV100_S        0.0       41C        0.0      0%       0%      15.7G        0M            0           ok           -
                    2 TeslaV100_S        0.0       39C        0.0      0%       0%      15.7G        0M            0           ok           -
                    3 TeslaV100_S        0.0       42C        0.0      0%       0%      15.7G        0M            0           ok           -
(base) [gargiulo@ems1 ~]$
```

Interessante notare la colonna `gpu_mode`, la quale se assume valore pari a 3, significa che quella gpu è occupata in maniera esclusiva da un utente.

Ovviamente, per i fini di questo documento, il comando di maggiore interesse è quello per sottomettere un job, `bsub` e, in particolare, come controllare le caratteristiche hardware che gli vengono riservate dal gestore: cpu, ram, gpu, etc. Tutti questi parametri sono configurabili e tutte le informazioni di dettaglio si

possono trovare sulla documentazione[2]. Di seguito, verranno mostrati solo quelli che potrebbero essere di interesse attraverso la descrizione di alcune soluzioni associate a scenari tipo.

Scenario 1: Sessione Interattiva

In questo caso si vuole accedere ad un nodo per poter eseguire dei comandi o lanciare dei programmi con una sessione interattiva. Per poterlo fare basta eseguire il comando

`bsub -Is /bin/bash` (l'opzione -Is crea uno pseudo-terminale)

con questo comando però non si ha a disposizione nessuna GPU, per avere anche l'accesso alle GPU bisogna modificare il comando come segue:

`bsub -gpu "num=1" -Is /bin/bash` (in questo caso si è specificato al gestore che si vuole usare una sola gpu)

Se si volesse specificare l'uso di una GPU usando solo 4GB di RAM, allora bisognerebbe modificare il comando come segue:

```
bsub -gpu "num=1:gmem=4G" -Is /bin/bash
```

Se a questo, si volesse aggiungere la possibilità di avere una GPU in maniera esclusiva ad esempio per misurare le performance di un test, indipendentemente da eventuali interferenze da parte di altri utenti del cluster, si potrebbe modificare il comando come segue:

```
bsub -gpu "num=1:mode=exclusive_process" -Is /bin/bash
```

Si rimanda alla documentazione sulla gestione delle GPU[3] per avere informazioni complete. Riporto di seguito solo la sinossi delle opzioni del parametro `-gpu`:

bsub -gpu - | **"[num=***num_gpus***[/task** | **host]] [:mode=shared** | **exclusive_process]**
[:mps=yes[,shared][,nocvd] | **no** | **per_socket[,shared][,nocvd]** | **per_gpu[,shared][,nocvd]]**
[:j_exclusive=yes | **no]** [**:aff=yes** | **no]** [**:block=yes** | **no]** [**:gpack=yes** | **no]**
[:gmodel=*model_name***[-***mem_size***]]** [**:gtile=***tile_num***|'!']** [**:gmem=***mem_value***]** [**:nvlink=yes]"**

Scenario 2: Sottomissione di un JOB e gestione degli output.

In questo scenario si vuole sottomettere un job ed analizzarne gli output, sia quelli "normali" che quelli generati in caso di errore. Il gestore permette di redirigere queste due tipologie di output su file di testo separati. Per esempio, si immagini di volere eseguire il programma `test.sh` fatto nel seguente modo:

```
#!/bin/bash
. ~/.bashrc
echo Ciao Mondo
```

Per sottomettere il job al gestore e intercettare gli output si potrebbe eseguire il comando:

```
bsub -o test.out -e test.err "bash test.sh"
```

[2] https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_command_ref/bsub.man_top.1.html
[3] https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_command_ref/bsub.gpu.1.html

in questo caso ci si è messi nell'ipotesi di trovarsi nella cartella dove è posizionato il file `test.sh`. Non essendo stati specificati i percorsi assoluti, dei file di output, `test.out` e `test.err` saranno creati nella cartella nella quale è stato lanciato il comando `bsub`.

Di seguito viene mostrato il contenuto del file `test.out` (non essendoci errori, il file `test.err` è chiaramente vuoto)

```
Ciao Mondo

------------------------------------------------------------
Sender: LSF System <lsfadmin@ben06>
Subject: Job 16471: <bash test.sh> in cluster <ben> Done

Job <bash test.sh> was submitted from host <ems1> by user <gargiulo> in cluster <ben> at Wed Nov  4 18:05:13 2020
Job was executed on host(s) <ben06>, in queue <normal>, as user <gargiulo> in cluster <ben> at Wed Nov  4 18:05:14 2020
</home/users/gargiulo> was used as the home directory.
</home/users/gargiulo> was used as the working directory.
Started at Wed Nov  4 18:05:14 2020
Terminated at Wed Nov  4 18:05:16 2020
Results reported at Wed Nov  4 18:05:16 2020

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
bash test.sh
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time :                                   0.57 sec.
    Max Memory :                                 17 MB
    Average Memory :                             12.67 MB
    Total Requested Memory :                     -
    Delta Memory :                               -
    Max Swap :                                   -
    Max Processes :                              5
    Max Threads :                                7
    Run time :                                   1 sec.
    Turnaround time :                            3 sec.

The output (if any) is above this job summary.



PS:

Read file <test.err> for stderr output of this job.

test.out (END)
```

Si noti come, la prima parte del file contiene l'output del programma `Ciao Mondo`, dopo di che è presente una parte descrittiva sul job effettuato da cui è possibile estrarre tutta una serie di informazioni utili, tra le quali: il tempo di processore, l'occupazione di memoria e altro.

Anche in questo caso è possibile usare moltre altre opzioni come quelle per l'uso delle GPU.

Scenario 3: Uccidere i job

Si supponga di aver avviato un processo che non funzioni correttamente, per poterlo fermare, LSF mette a disposizione il comando `bkill`, a tal fine basta prima identificare il `JOBID` del job incriminato e quindi procedere all'eliminazione come di seguito:

```
(base) [gargiulo@ems1 ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
16463   gargiul RUN   interactiv ems1        ben02       /bin/bash  Nov  2 18:19
(base) [gargiulo@ems1 ~]$ bkill 16463
```

Nel caso, più complesso, di aver generato centinaia di processi che non funzionano correttamente, si può utilizzare il comando `bkill 0` per eliminarli tutti.

4. La gestione delle repository

Nel cluster, la modalità classica di operare è attraverso ambienti conda[4] (conda environment). In un ambiente conda è possibile installare tutti i pacchetti necessari per sviluppare o usare un programma. In particolare, la prima cosa da configurare sono i canali.
Per l'architettura POWERPC, la IBM, ha definito dei canali propri con le distribuzioni dei pacchetti specificatamente modificate per ottimizzare l'architettura del cluster.

Uno si questi canali è:

```
https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/
```

ed è configurabile nel sistema attraverso il seguente comando (n.b. di default sulla piattaforma è già configurato quindi non è necessario effettuare questo passaggio):

```
conda config --prepend channels \
https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/
```

A questo punto è possibile poter creare un ambiente conda con tutto il necessario per poter operare. Di seguito viene riportato un estratto della documentazione presente sul sito della IBM[5].
La sintassi per creare ed attivare l'ambiente è:

```
conda create --name <environment name> python=<python version>
conda activate <environment name>
```

Per esempio, per creare un ambiente di nome wmlce_env con la versione di Python 3.6, si deve eseguire il comando.

```
conda create --name wmlce_env python=3.6
conda activate wmlce_env
```

Gli ambienti sono collezioni private di software che risiedono nella home directory dell'utente. La directory che contiene gli ambienti personali di default di ciascun utente è: ~/.conda/envs

A questo punto, per installare all'interno del proprio ambiente tutte le risorse che IBM ha ottimizzato per il deep learning, machine learning, big data analytics e alle quali la IBM ha dato nome commerciale Watson Machine Learning Community Edition (WMLCE), basta eseguire il comando:

```
conda install powerai
```

Sulla sito della IBM, sono spiegati nel dettaglio configurazioni e opzioni "particolari" che esulano dallo scopo di questo documento. Si rimanda il lettore alla documentazione per maggiori dettagli.

Al fine di fornire degli strumenti immediatamente eseguibili sul cluster in esame, di seguito si riportano alcuni casi di interesse e comandi utili.

Per poter capire quali ambienti sono fruibili da subito, basta eseguire il comando:

[4] https://docs.conda.io/projects/conda/en/latest/index.html
[5] https://www.ibm.com/support/knowledgecenter/SS5SF7_1.7.0/navigation/welcome.html

```
(base) [gargiulo@ems1 ~]$ conda-env list
# conda environments:
#
allennlp                 /home/users/gargiulo/.conda/envs/allennlp
flair_env                /home/users/gargiulo/.conda/envs/flair_env
hf                       /home/users/gargiulo/.conda/envs/hf
poweraiFG                /home/users/gargiulo/.conda/envs/poweraiFG
transformers             /home/users/gargiulo/.conda/envs/transformers
wmlce                    /home/users/gargiulo/.conda/envs/wmlce
wmlce_1_7                /home/users/gargiulo/.conda/envs/wmlce_1_7
wmlce_2                  /home/users/gargiulo/.conda/envs/wmlce_2
wmlce_3                  /home/users/gargiulo/.conda/envs/wmlce_3
base                  *  /opt/anaconda3
wmlce16                  /opt/anaconda3/envs/wmlce16

(base) [gargiulo@ems1 ~]$
```

In questo caso, si possono notare degli ambienti associati solo all'utente "gargiulo", mentre ce ne sono due, validi per tutti gli utenti e sono: base e wmlce16.

A questo punto, è possibile attivare uno degli environment (ad esempio wmlce16) e verificare quali pacchetti sono disponibili in esso, attraverso i comandi:

```
(base) [gargiulo@ems1 ~]$ conda activate wmlce16
(wmlce16) [gargiulo@ems1 ~]$ conda list
# packages in environment at /opt/anaconda3/envs/wmlce16:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                         main
_py-xgboost-mutex         1.0            gpu_605.g99281e0    https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda
_pytorch_select           2.0            gpu_20251.ga479d1e    https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda
_tflow_select             2.1.0          gpu_861.gea19599    https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda
absl-py                   0.7.1                    py37_0
apex                      0.1.0_1.6.2    py37_596.g1eb5c77    https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda
asn1crypto                1.3.0                    py37_0
astor                     0.7.1                    py37_0
atomicwrites              1.3.0                    py37_1
attrs                     19.3.0                     py_0
blas                      1.0                   openblas
bokeh                     1.4.0                    py37_0
boost                     1.67.0                   py37_4
bzip2                     1.0.8                h7b6447c_0
```

con il secondo comando (`conda list`) è possibile elencare tutti i pacchetti contenuti nell'ambiente attivo, specificando, per ognuno di essi: il nome del pacchetto, la versione, i dettagli sulla versione (build) ed il canale dalla quale è stato scaricato il pacchetto. Inoltre, una volta attivato l'ambiente, la scritta iniziale tra parentesi tonde cambia, specificando all'utente quale ambiente è attivo da `(base)` a `(wmlce16)`

Per disattivare l'ambiente basta eseguire il comando:

```
conda deactivate
```

Per installare, aggiornare o cancellare dei pacchetti, la cosa più corretta è clonare l'ambiente desiderato in uno proprio attraverso il comando seguente:

```
conda create --name myclone --clone myenv
```

Passiamo a come installare un pacchetto, si supponga di avere la necessità di installare "scipy".

```
(base) [gargiulo@ems1 ~]$ conda activate wmlce_2
(wmlce_2) [gargiulo@ems1 ~]$ conda search scipy
Loading channels: done
# Name                       Version           Build  Channel
scipy                          1.0.0  py27h1093705_0  pkgs/main
scipy                          1.0.0  py35h1093705_0  pkgs/main
scipy                          1.0.0  py36h1093705_0  pkgs/main
scipy                          1.0.1  py27h9d22d0a_0  pkgs/main
scipy                          1.0.1  py35h9d22d0a_0  pkgs/main
scipy                          1.0.1  py36h9d22d0a_0  pkgs/main
scipy                          1.1.0  py27h9c1e066_0  pkgs/main
scipy                          1.1.0  py27h9d22d0a_0  pkgs/main
scipy                          1.1.0  py35h9c1e066_0  pkgs/main
scipy                          1.1.0  py35h9d22d0a_0  pkgs/main
scipy                          1.1.0  py36h9c1e066_0  pkgs/main
scipy                          1.1.0  py36h9d22d0a_0  pkgs/main
scipy                          1.1.0  py37h9c1e066_0  pkgs/main
scipy                          1.3.0  py36h807e534_0  conda-forge
scipy                          1.3.0  py36he2b7bc3_0  pkgs/main
scipy                          1.3.0  py37h807e534_0  conda-forge
```

Prima di tutto è necessario attivare un environment locale dell'utente, in questo caso, è stato scelto l'ambiente utente `wmlce_2`, quindi si cerca il pacchetto e si passa all'installazione della versione desiderata.

Attualmente sui canali della IBM la versione più aggiornata è la 1.5.3 disponibile su conda-forge

```
scipy  1.5.3  py39h6d1a728_0  conda-forge
```

nell'ambiente di esempio `wmlce_2` al momento è installata una versione non aggiornatissima di scipy, come si nota dal seguente screenshot:

```
(wmlce_2) [gargiulo@ems1 ~]$ conda list | grep scipy
scipy                     1.3.1           py37he2b7bc3_0
(wmlce_2) [gargiulo@ems1 ~]$
```

se si fosse voluto installare una versione specifica del pacchetto, si sarebbe dovuto eseguire il comando:

```
conda install scipy=1.5.3
```

Invece, per aggiornarlo, bisogna eseguire il comando:

```
conda update scipy
```

Nel caso in cui sia necessario rimuoverlo, basta eseguire il comando:

```
conda remove scipy
```

In un ambiente conda è possibile installare dei pacchetti anche attraverso il comando `pip`[6] di python. Pip (acronimo di Pip Installs Packages) è uno strumento a linea di comando che permette di installare dei pacchetti in maniera analoga a `conda`. I pacchetti vengono reperiti da PyPI (Python Package Index). Quest'ultimo è un repository globale, dove vengono archiviati migliaia di progetti e programmi relativi a Python, per essere quindi gestiti e organizzati in base alle versioni dei pacchetti e alle loro dipendenze. Si rimanda alla documentazione, per una lista esaustiva di tutte le opzioni usabili per questo comando.

[6] https://pypi.org/project/pip/

```
(wmlce_2) [gargiulo@ems1 ~]$ pip search scipy | grep scipy
scipy (1.5.3)                  - SciPy: Scientific Library for Python
```

In questo caso, la versione più aggiornata di `scipy` sul canale conda, corrisponde con quella più aggiornata su PyPI, ma purtroppo questo non è sempre vero dato che in questo canale non sono sempre disponibili le ultime versioni compilate per la piattaforma POWERPC.

5. Simulare una WorkStation

In questa sezione sarà descritta una modalità operativa per usare il cluster, ossia come simulare una workstation con interfaccia grafica. A tal fine sono stati predisposti degli ambienti ad hoc, per poter rapidamente configurare la postazione. Di seguito verranno mostrati gli step necessari per abilitare questa funzionalità.

5.1 La configurazione di VNC -- lato server.

La prima operazione è attivare un server VNC[7] (Virtual Network Computing) sul cluster, a tal fine, una volta collegati al server tramite ssh, va invocato il comando:

```
vncserver
```

La prima volta che si esegue il comando verranno richieste delle password, una per la modalità (normale) ed un'altra per la modalità ospite in sola lettura.

Per capire se l'operazione è andata a buon fine, basta verificare se c'è un'istanza del server VNC attiva come nel seguente esempio:

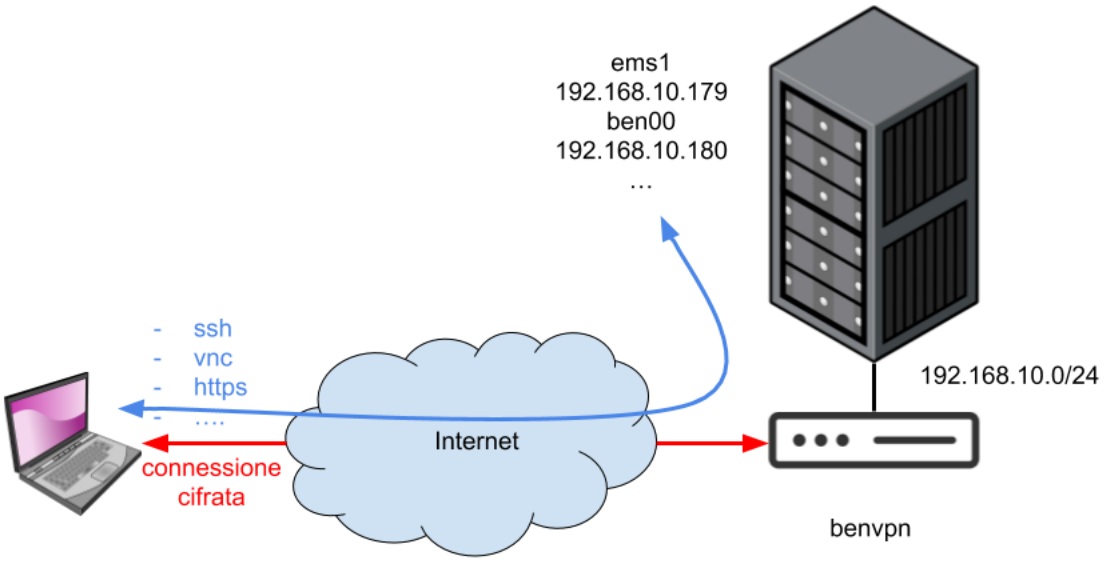
```
(wmlce_2) [gargiulo@ems1 ~]$ vncserver -list

TigerVNC server sessions:

X DISPLAY #     PROCESS ID
:1              53314
(wmlce_2) [gargiulo@ems1 ~]$
```

In questo caso, dall'output di questo comando possiamo recuperare due informazioni fondamentali: il numero del display "1", ed il process ID "53314". E' possibile cambiare le password in ogni momento attraverso il comando `vncpasswd`. Tutti i file di configurazione, i log, i pid, le password cifrate si trovano nella cartella `~/.vnc` (dove il simbolo ~ corrisponde alla home directory dell'utente).

5.2 (a) La configurazione di VNC (tramite VPN) -- lato client.

Per poter usare VNC viewer sul client basta avviare una connessione VPN[8] che permetterà all'utente di collegarsi al `vncserver` direttamente dal frontend (`ems1`) che ha un indirizzo IP `192.168.10.179`.

[7] https://archive.realvnc.com/products/vnc/documentation/4.1/unix/man/vncserver.html
[8] https://www.potaroo.net/papers/1998-3-vpn/vpn.pdf

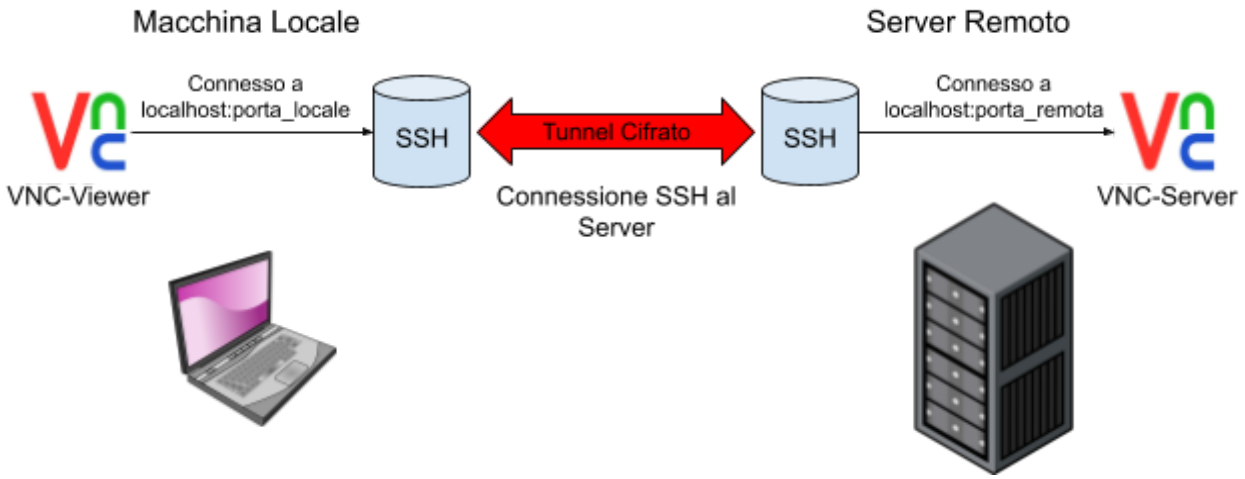


Come si può notare dalla figura in alto, la VPN simula la presenza della macchina locale direttamente sulla rete LAN del server remoto. In questo caso a valle del collegamento non bisogna fare null'altro, come detto in precedenza, che collegarsi opportunamente alla macchina sulla quale è in esecuzione il server VNC.

5.2 (b) La configurazione di VNC (tramite tunnel SSH) -- lato client.

La seconda possibilità per usare un VNC viewer sul client è attraverso un tunnel SSH. A tal fine, si deve avviare una sessione SSH[9] con l'opzione -L che crea un tunnel SSH per redirigere il traffico di una specifica porta del server sull'interfaccia localhost della propria macchina. Nella figura seguente viene spiegata meglio questa operazione: sul lato sinistro viene rappresentata la macchina locale dell'utente, l'applicativo (VNC viewer), che si collega al canale `localhost:porta_locale`; mentre sul lato destro è rappresentato il server remoto, l'applicativo (VNC server), che si collega al canale `localhost:porta_remota`. Come detto in precedenza, il canale è stato creato attraverso attraverso una connessione SSH cifrata (freccia rossa bidirezionale centrale). Si noti che i due localhost fanno riferimenti a due *loopback*[10] diversi rispettivamente del client e del server.

[9] https://man.openbsd.org/ssh
[10] https://tools.ietf.org/html/rfc3330



La sintassi generale del comando SSH per la creazione del tunnel e la predisposizione delle porte, sia lato client che lato server, è la seguente:

```
ssh -L porta_locale:indirizzo_remoto:porta_remota username@server_ssh
```

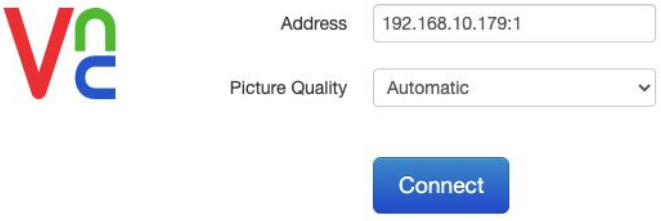
nel caso dell'esempio, il comando si sostanzia nel seguente modo:

```
ssh -L 5901:localhost:5901 gargiulo@ben.na.icar.cnr.it
```

Si noti come è stata scelta come porta locale la 5901[11] e come porta remota sempre la 5901, quest'ultima scelta è però obbligata e dipende dal #DISPLAY che viene dato dal server all'atto dell'esecuzione del comanda `vncserver`, in questo caso il #DISPLAY era 1 e quindi il numero di porta è 5901, se fosse stato 4 il numero di porta da mettere sarebbe stato 5904. In generale, dato il #DISPLAY, il numero di porta remota da inserire si calcola come 5900 + #DISPLAY.

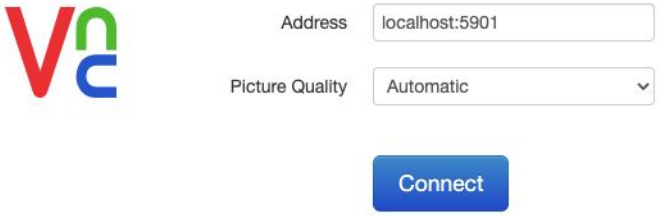
5.3 La configurazione di VNC -- Uso di un client e creazione di una sessione di lavoro.

Una volta creata o a) la connessione VPN o b) il tunnel SSH, è possibile usare un qualunque client VNC per avviare la sessione di lavoro, nel caso degli esempi in basso VNC VIEWER for GOOGLE CHROME.

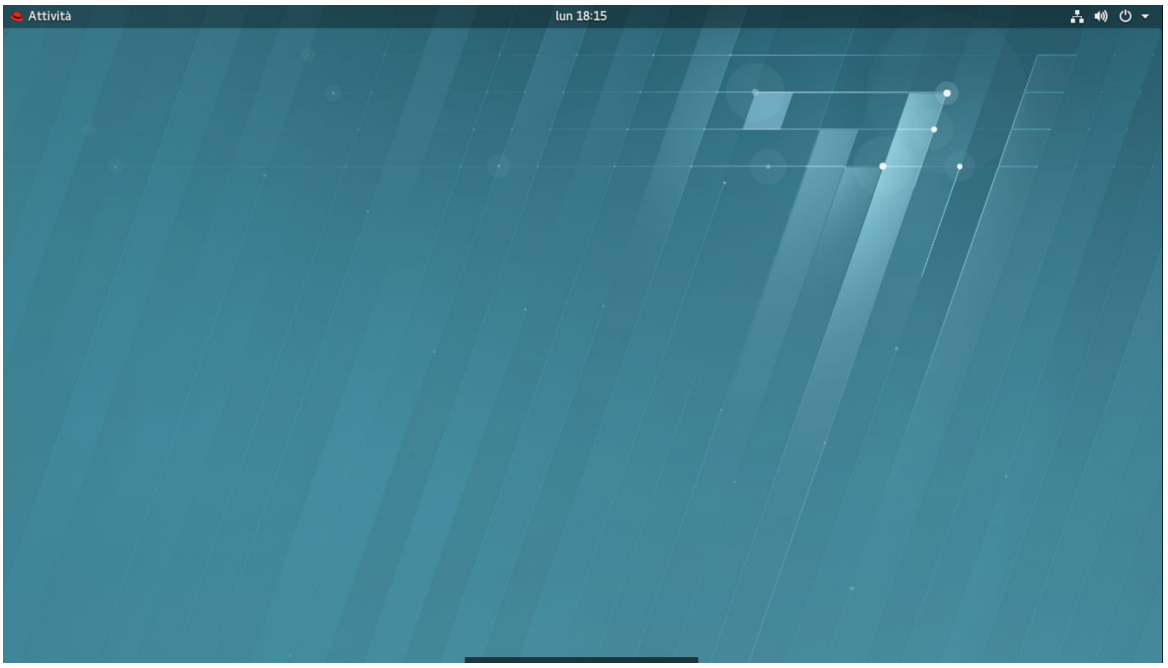


a) **Connessione VNC nel caso di canale VPN.** Bisogna inserire l'indirizzo IP del server ems1 che ospita il processo VNCSERVER specificando il #DISPLAY ossia `192.168.10.179:#DISPLAY`)

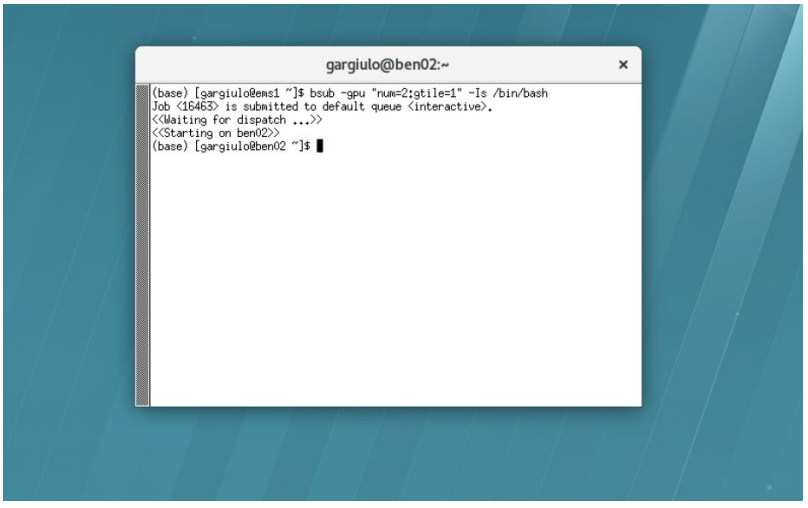
[11] Il numero di porta 5901 è quello standard di VNC, ma nulla vieta di sceglierne uno completamente arbitrario, a patto di non sovrapporlo con qualcuno già attivo sulla macchina.



b) **Connessione VNC nel caso di canale SSH.** In questo caso, bisogna inserire come indirizzo `localhost:porta_locale,` per accedere all'ambiente grafico preimpostato.



L'interfaccia grafica appare come in figura, in esso è stato installato un ambiente `gnome` volutamente reso più leggero eliminando animazioni e superflui effetti grafici. Per avviare l'ambiente di sviluppo, considerando che questa interfaccia grafica gira sul frontend (ems1) e che **non deve** essere utilizzato in **nessun caso** per fare calcolo, bisogna eseguire in un terminale un batch interattivo con le caratteristiche hardware necessarie per la realizzazione ed il debug dell'applicativo.



Nella screenshot di esempio in alto, dopo aver avviato xterm è stato eseguito il comando:

```
bsub -gpu "num=2:gtile=1" -Is /bin/bash
```

In questo caso sono state richieste due GPU. Se fosse stata necessaria una sola GPU, ma in maniera esclusiva, escludendo l'accesso ad altri utenti, si sarebbe dovuto usare il comando:

```
bsub -gpu "num=1:gtile=1:mode=exclusive_process" -Is /bin/bash
```

Si rimanda alla sezione 3 dove si parla del gestore dei processi LSF per maggiori informazioni.

Una volta avviato il job si potrà verificare che è running eseguendo i seguenti comandi (`bqueues` e `bjobs`):

```
francescogrg@MacBook-Francesco .ssh % ssh -L 5901:localhost:5901 gargiulo@ben.na.icar.cnr.it
Last login: Mon Nov  2 17:01:56 2020 from host-79-23-9-42.retail.telecomitalia.it
(base) [gargiulo@ems1 ~]$ bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
wmla            100  Open:Active       -    -    -    -     0     0     0     0
admin            50  Open:Active       -    -    -    -     0     0     0     0
priority         43  Open:Active       -    -    -    -     0     0     0     0
short            35  Open:Active       -    -    -    -     0     0     0     0
dataq            33  Open:Active       -    -    -    -     0     0     0     0
normal           30  Open:Active       -    -    -    -     0     0     0     0
interactive      30  Open:Active       -    -    -    -     1     0     1     0
(base) [gargiulo@ems1 ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
16463   gargiul RUN   interactiv ems1        ben02       /bin/bash  Nov  2 18:19
(base) [gargiulo@ems1 ~]$
```

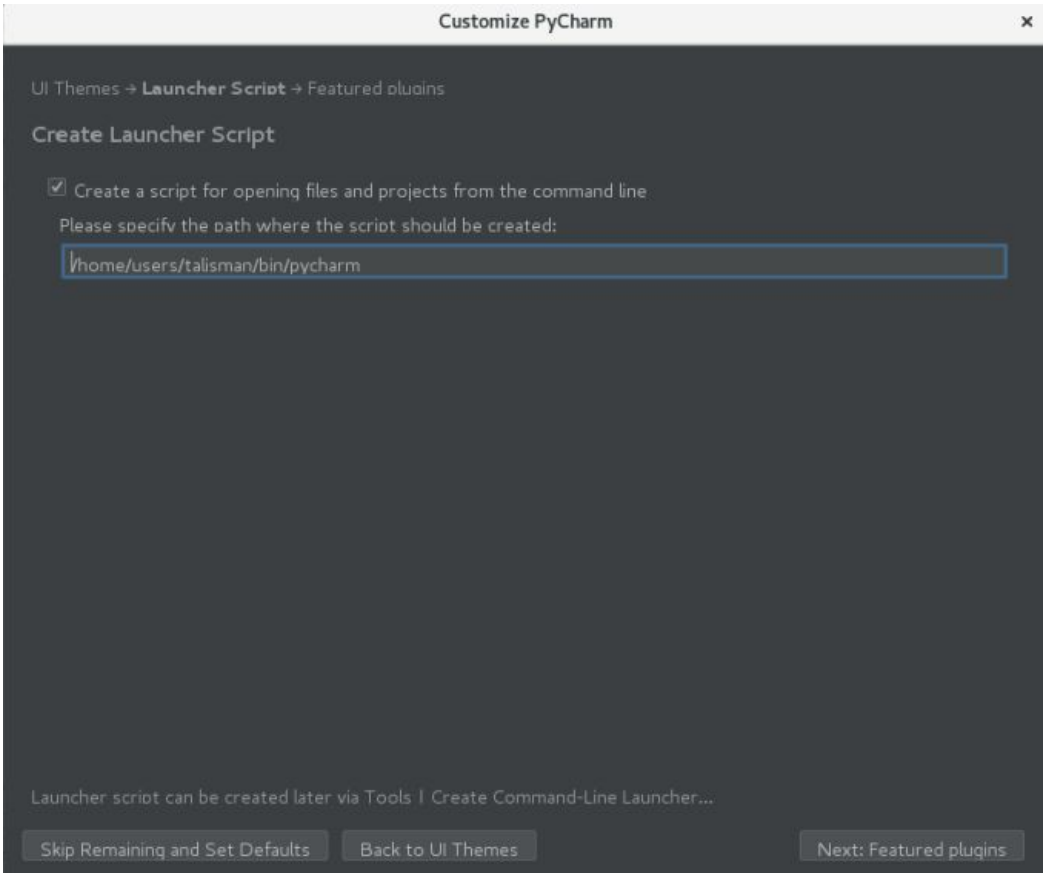
Si noti come il job viene inserito nella coda "interactive" e resterà attivo anche se la connessione dovesse venir meno per qualsiasi motivo, ottenendo un ambiente persistente nel tempo e stabile.

A questo punto, è possibile avviare l'ambiente IDE desiderato, ad esempio pycharm[12], attraverso il comando:

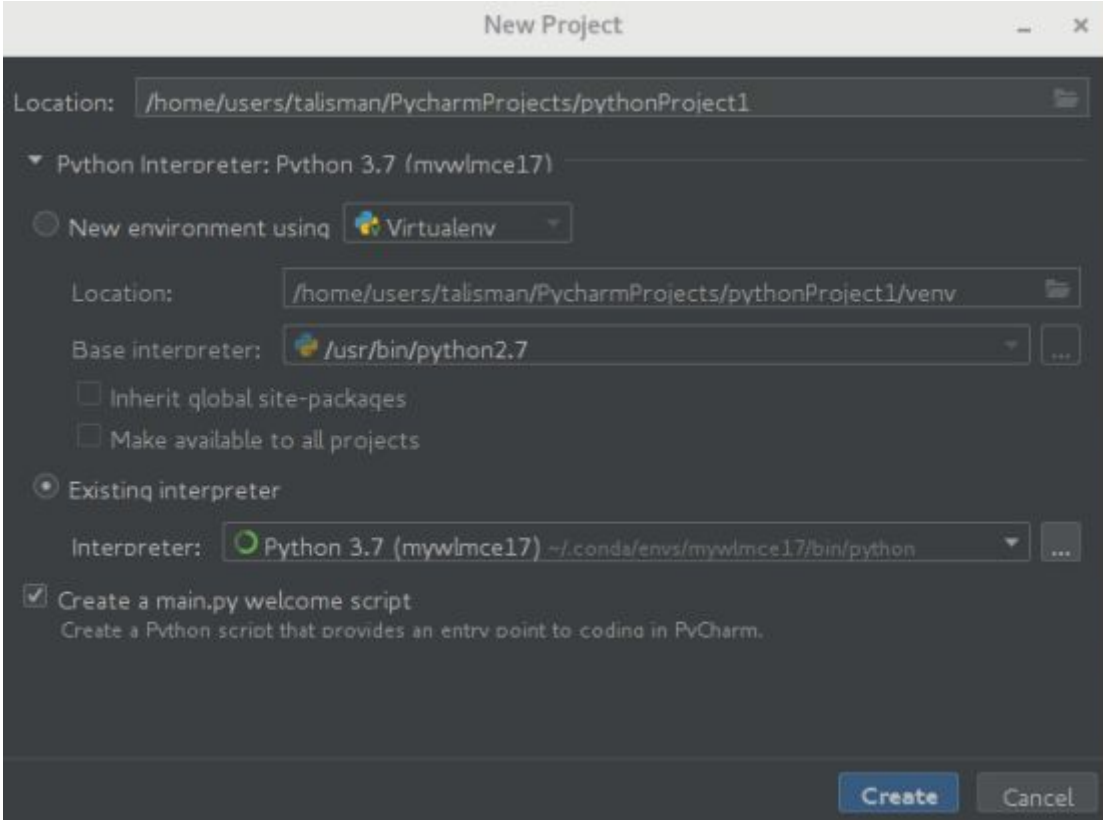
```
(base) [gargiulo@ben02 ~]$ /opt/software/pycharm/bin/pycharm.sh
```

La prima volta che viene avviato l'ambiente è possibile configurare un lanciatore in maniera da agevolare i successivi avvii. Si consideri l'esempio mostrato nella figura in basso, alla comparsa della finestra "*Create Launcher Script*" bisogna indicare il nome del lanciatore che si vuole creare (ad esempio `pycharm`) avendo l'accortezza di inserirlo in un percorso che si trova nel `$PATH` di sistema. In altre parole, va specificato il percorso: `/home/users/<<nome_utente>>/bin/pycharm`. In questo modo, dal secondo avvio, sarà possibile eseguire direttamente il comando `pycharm` da terminale.

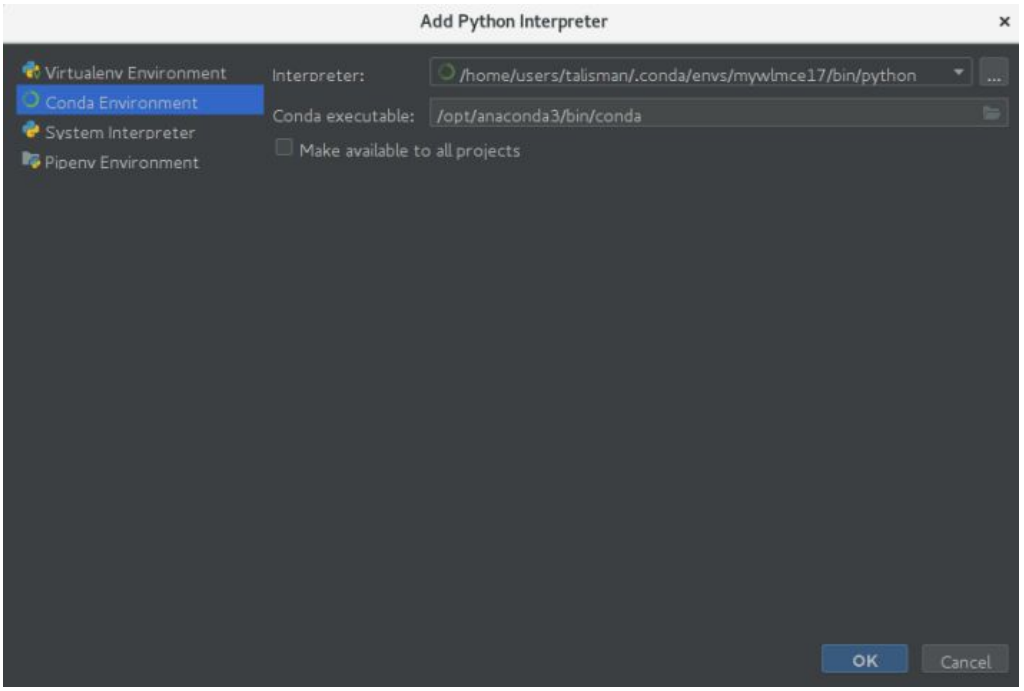
[12] https://www.jetbrains.com/pycharm/



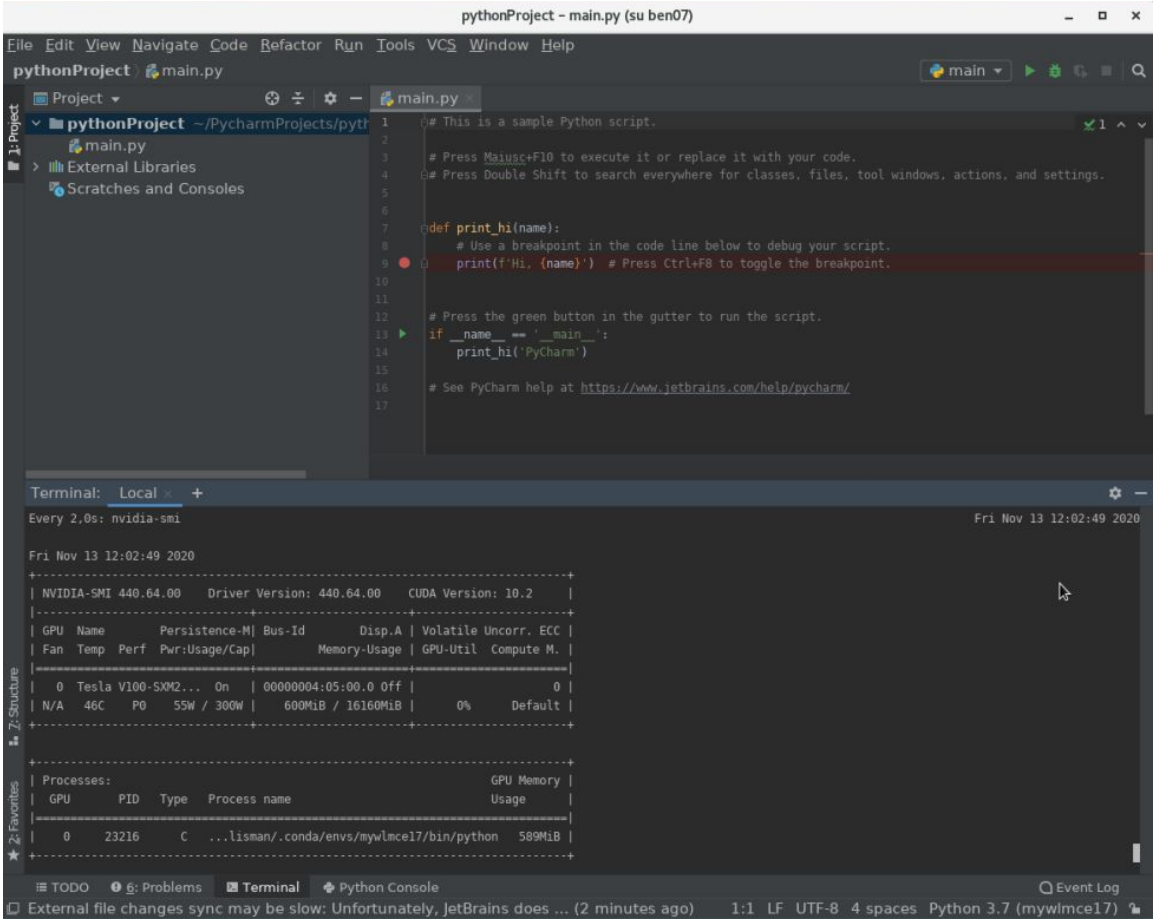
Una volta configurato l'ambiente, sarà possibile creare un nuovo progetto, come nella figura di esempio in basso, specificando l'ambiente conda di riferimento, in questo caso `mywmlce17`.



Nel seguente screenshot viene mostrato che bisogna specificare che si userà un interprete appartenente ad un ambiente *conda*. Si noti che è opportuno selezionare un ambiente "modificabile" tra quelli che si trovano in `~/.conda/envs`.



Ultimate queste procedure, ci si troverà nell'ambiente di sviluppo, come da figura in basso.



6. Pianificare una sessione di test

Una volta ottenuta un'applicazione stabile, per fare il tuning dei parametri, normalmente sono necessari molti test e valutazioni. La batteria di test va attentamente pianificata e lanciata sulle risorse che mette a disposizione il cluster, chiaramente in modalità batch ed in parallelo.
A tal fine vanno preparati degli script *BASH*[13]. Di seguito sarà descritta una metodologia che potrà essere usata come esempio per le proprie finalità.

I passi che potrebbero essere seguiti per lanciare una batteria di test, sono essenzialmente tre:

1. **Adattamento del programma principale**: si ipotizzi che il punto di ingresso del programma sia nel file `main.py`, in questo file devono essere predisposte delle variabili di ingresso per ognuno dei parametri da valutare in fase di test. Di seguito una possibile modalità per intercettare due parametri esterni in ingresso al programma: `var1` e `var2`.

```
[..]
def run():
    parser = argparse.ArgumentParser(description='Variabili di ingresso')
    parser.add_argument("-var1", dest="var1", default=0.0, help="Variabile a")
    parser.add_argument("-var2", dest="var2", default=0.0, help="Variabile b")
    args = parser.parse_args()
[..]
```

2. **Script di RUN**: creazione di uno script che invoca il programma principale e che configura l'ambiente corretto di esecuzione, si supponga si chiami `run.sh`. Si noti come la prima e la seconda istruzione, sono funzionali a ricreare lo stesso contesto nel quale abbiamo realizzato il programma: variabili di ambiente, e altro. Quindi si attiva l'ambiente conda desiderato (linea 3) e ci si posiziona nella cartella dove è presente il file `main.py` (linea 4). Infine (linea 5) si esegue il programma avendo cura di parametrizzare le variabili esterne, in questo caso, rinominate $1 e $2.

```
#!/bin/bash
. ~/.bashrc
conda activate wmlce16
cd ~/ApplicationFolder #Cartella dove si trova il file main.py
python main.py -var1 $1 -var2 $2
```

3. **Script LSF:** creazione di uno script che esegua, tramite il gestore di job LSF, il `run.sh` precedentemente realizzato con tutte le configurazioni delle variabili che si ritengono necessarie e con la configurazione hardware ipotizzata; si supponga di identificare questo script con il nome `test_bsub.sh`. Ad esempio, nello script in basso viene prima definita una variabile con le specifiche hardware richieste (`gpu_option`), subito dopo un'altra variable `range_weights` con la lista dei valori da testare. Infine, attraverso due cicli `for` innestati, uno per ciascuna variabile (`var1` e `var2`), viene lanciato lo script `run.sh` con tutte le possibili configurazioni di variabili. In questo caso essendoci 2 variabili e, per ognuna di esse, 3 possibili valori, si avranno un totale di 9 test.

[13] https://www.gnu.org/software/bash/manual/bash.html

```
#!/bin/bash
gpu_option="num=1:gtile=1:gmem=7.5G"

range_weights=(0 0.5 1)

for var1 in ${range_weights[@]}
do
for var2 in ${range_weights[@]}
do
bsub -gpu ${gpu_option} -o output/test_${var1}_${var2}.out -e
output/test_${var1}_${var2}.err "bash run.sh ${var1} ${var2}"
done
done
```

A questo punto sarà possibile monitorare lo stato di esecuzione dei test attraverso le metodiche precedentemente descritte nella sezione 3.

Conclusioni

In questo rapporto tecnico si è presentata la nuova architettura hardware IBM di cui l'ICAR-CNR si è munito per le sue attività di ricerca. Sono state presentate alcune delle modalità operative per l'utilizzo di questa piattaforma, in futuro, laddove gli scenari di utilizzo e le necessità dovessero mutare, si potrebbe pensare di aggiungere o modificare, anche radicalmente, le modalità di fruizione del cluster da parte degli utenti. Ad esempio, si potrebbe cambiare il gestore di processi ed usarne uno più complesso come *IBM Watson® Machine Learning Accelerator* (WMLA) o cambiare il paradigma per incapsulare gli applicativi, per esempio facendo uso dei docker. Si potrebbe pensare di ottimizzare la parte di modellistica fornendo un front-end basato su Jupiter/Anaconda e molto altro ancora.

Riferimenti

[1] Documentazione del gestore dei processi LSF disponibile sul sito della IBM https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_welcome/lsf_welcome.html

[2] Documentazione del programma VNC https://archive.realvnc.com/products/vnc/documentation/4.1/unix/man/vncserver.html

[3] Documentazione del tool SSH https://man.openbsd.org/ssh

[4] Documentazione della VPN https://www.potaroo.net/papers/1998-3-vpn/vpn.pdf

[5] Riferimento all'ambiente ide PYCHARM https://www.jetbrains.com/pycharm/

[6] Documentazione del BASH script https://www.gnu.org/software/bash/manual/bash.html

[7] Documentazione del gestore dei pacchetti CONDA https://docs.conda.io/projects/conda/en/latest/index.html

[8] Documentazione del gestore dei pacchetti PIP https://pypi.org/project/pip/