Manuale per l'utilizzo della DGX-A100 del DIAG

Il presente documento contiene le linee guida per l'utilizzo della DGX-A100 in uso presso il DIAG-Sapienza.

Caratteristiche della DGX-A100

La DGX-A100 è un sistema progettato da NVIDIA per l'addestramento di Deep Neural Networks e il calcolo parallelo su GPU. Le sue caratteristiche hardware sono:

- 8 schede grafiche NVIDIA A100 ognuna fornita di 40GB di RAM
- 1TB di RAM di sistema
- 2 processori AMD Rome 7742 per un totale di 128 cores
- 15TB di storage interno per i dati

La macchina è connessa alla rete del dipartimento con due interfacce: una ethernet 1GBps utilizzabile per gestione remota (ssh o JupyterHub) e una scheda Mellanox con collegamento in fibra ottica a 20GBps utilizzabile per collegare la macchina a storage esterni (NAS) tramite protocollo NFS.

La macchina è collocata nel CED del DIAG.

DGX-A100 nel suo rack presso il CED

Indirizzi IP:

Ssh/JupyterHub 192.168.60.200 (ssh su porta 22, e JupyterHub su porta 8000)

Linee Guida Generali

Al fine di massimizzare e rendere più agevole l'uso della DGX da parte dei vari gruppi di ricerca del dipartimento, la stessa potrà essere utilizzata esclusivamente per esperimenti su modelli già rifiniti e precedentemente testati; **in particolare la macchina non va usata per la prototipazione e lo sviluppo**. In dettaglio, le motivazioni di questa scelta sono:

1) Lo sviluppo e il raffinamento di un modello richiede spesso un lungo processo di trial and error che occuperebbe le risorse della macchina con micro-burst di computazione. Questo porta ad avere una macchina formalmente occupata da un utente senza che essa esegua calcoli, diminuendo in modo evidente la sua percentuale di reale utilizzo. Per la prototipazione degli esperimenti è più sensato utilizzare strumenti quali Colab o altre risorse a disposizione del singolo gruppo di ricerca per arrivare ad un modello finito.
2) Lo spazio disco della macchina non è abbastanza grande da permettere la stanzialità di molti dataset sullo stesso.

La linea guida generale è quindi la seguente:

1. Ogni utilizzatore della macchina crea e rifinisce il suo modello e i dati su risorse private.
2. Una volta che il modello è finito esso prenota la macchina, o parte di essa, (vedere sezione prenotazioni).
3. All'interno del suo slot temporale prenotato carica il modello e i dati di test sulla macchina, facendo attenzione a non usare GPU non prenotate, ed esegue gli esperimenti finali.
4. Al termine degli stessi (o del periodo di uso prenotato) colleziona i suoi risultati e libera lo spazio della macchina dai dataset.

Containerizzazione

Al fine di dare la possibilità ad ogni utente di installare l'ambiente di esecuzione di cui ha bisogno (ad esempio, una particolare versione di pytorch o tensorflow) l'esecuzione di esperimenti sulla macchina avviene sempre e soltanto in un container privato dell'utente. Dal punto di vista dell'utilizzo, i container sono assimilabili a delle macchine virtuali che separano gli ambienti utente permettendo a ciascuno di installare le librerie e le applicazioni di cui ha bisogno senza influenzare gli ambienti di sperimentazione usati da altri.

Sulla macchina è predisposto l'ambiente di containerizzazione Docker (https://docs.docker.com/). L'utente quindi deve creare una propria immagine docker, oppure usare una immagine tra quelle già disponibili su internet (catologo immagini nvidia https://ngc.nvidia.com/catalog). E usare un container derivato da questa immagine per eseguire i propri esperimenti.

Utenze

La creazione delle utenze avviene tramite richiesta all'indirizzo XXXX. Le utenze sono assegnate in numero di una per **personale strutturato**, ma possono essere date in uso a personale non strutturato. Ogni strutturato rimane responsabile dell'uso corretto dell'utenza.

L'uso della macchina avviene tramite prenotazione. Ogni prenotazione permette di riservare una GPU della macchina.
Le GPU sono numerate con ID che vanno da 0 a 7. L'utente che prenota un sottoinsieme di queste è tenuto all'utilizzo esclusivo di questo sottoinsieme. Per consigli su come fare vedere la parte utilizzo della macchina.

> Ogni utente è tenuto a verificare che la macchina (e la rispettiva GPU) sia stata prenotata prima di caricare un modello su una GPU. **Questo è importante poiché mandare in overflow la RAM della GPU comporta l'interruzione degli esperimenti anche di chi aveva correttamente prenotato la risorsa.**

Sospensione e terminazione utenze

Un utenza può essere sospesa o terminata a causa delle violazioni delle regole di servizio al presente manuale.
Un utenza sarà terminata in maniera permanente in caso essa sia usata per attacchi (dimostrativi e non) verso altri utenti o terzi. Questo include attività di penetration testing.

Raggiungere la macchina usando la vpn

Per raggiungere la macchina dietro VPN si deve inviare una mail a Tiziana Toni chiedendo di abilitare il routing verso l'indirizzo 192.168.60.200. Altrimenti la macchina risulterà irraggiungibile.

Prenotazioni e regolamento

Per garantire un equo uso condiviso della macchina essa va prenotata usando il seguente foglio condiviso: Prenotazione DGX (accessibile con l'account diag). Le prenotazioni possono essere fatte solo da o a nome di personale strutturato (professori e/o ricercatori).

A regime questo foglio sarà sostituito da un applicativo web.
La prenotazione minima è di un giorno GPU.

Garantire la fairness

Per garantire un utilizzo delle risorse soddisfacente per tutti i docenti interessati è opportuno definire delle limitazioni alla quantità di risorse prenotabili. Le regole qui illustrate saranno considerate per una prima fase di utilizzo. A valle di questa, dopo una valutazione delle modalità d'uso osservate, saranno apportate le opportune modifiche per la fase due di operazione della macchina.

Crediti: Ogni strutturato ha un carnet di 32 giorni GPUs corrispondenti ad altrettanti crediti. La prenotazione di un giorno GPU consuma un credito. Questi crediti permettono, ad esempio, di prenotare tutte e 8 le GPU della macchina per 4 giorni consecutivi, o una singola GPU per 32 giorni, o qualsiasi altra combinazione tra numero di GPU e durata.
I crediti impiegati vengono riaccreditati al termine di un blocco di prenotazione. Un blocco di prenotazione è una prenotazione contigua della stessa GPU per un dato numero di giorni. Ad esempio supponiamo di prenotare la GPU 0 per 8 giorni consecutivi, dal giorno 1 al giorno 7. Questi 8 crediti saranno riaccreditati alla fine del giorno 7.

Il meccanismo di riaccredito permette ad un singolo utente di utilizzare la DGX illimitatamente se non c'è interesse da parte di altri utenti. Allo stesso tempo, fa si che in caso di maggiore richiesta, le GPU siano accessibili da tutti i richiedenti (in momenti diversi)

Prenotazione core: La DGX ha 128 core CPU che possono essere utilizzati per processamento classico parallelo. Visto che il timesharing sui core è gestito bene dal sistema operativo la prenotazione degli stessi non è regolata

(se non per il comune buonsenso). Si noti che ad uno slot di 30 core non possono essere assegnati dei core specifici, in questo caso l'utente si impegna a non lanciare più di 30 processi.
Da alcuni esperimenti effettuati l'utilizzo anche intensivo dei core CPU non impatta in maniera sensibile sul training di modelli, anche quando questi usano tutte e 8 le gpu.

Utilizzo della macchina tramite JupyterHub

Ad ogni utenza della macchina è automaticamente associato un account sul server JupyterHub in esecuzione sulla DGX. Il server è in ascolto sull'indirizzo: http://192.168.60.200:8000/.

Puntando un browser all'indirizzo di cui sopra l'utente si ritrova una schermata di login (le credenziali sono le stesse di SSH). Una volta loggato l'utente può selezionare una certa immagine docker tra quelle presenti.

Selezionata l'immagine il JupyterHub crea automaticamente un container docker associato alla stessa. In questo container è in esecuzione un server JupyterLab che viene mostrato all'utente. Questo server è quindi già containerizzato e in questo ambiente in cui l'utente ha a disposizione tramite interfaccia web:

- Una console ssh
- Un sistema per l'esecuzione di notebook python
- Un interfaccia visuale per il caricamento di file.

Questo ambiente è a totale disposizione dell'utente che può installare qualsiasi applicazione, è infatti un container compartimentato rispetto alla macchina.

L'utente di default del container è *jovyan* la password di root è *root*. La directory home nel container contiene una subdir *work* persistente, ossia i dati salvati in *work* sono gli unici per cui è garantita la permanenza sul sistema anche dopo il termine della sessione di lavoro o dopo il riavvio del server.

Questa directory *work* è sincronizzata automaticamente con il filesystem della DGX.

Sono predisposte due immagini:

- **pytorch-tf**: Contiene pytorch XX, tensorflow XX e CUDA 11.2 e altri pacchetti a supporto (numpy, scipy, ecc).

- **python-only**: Contiene soltanto CUDA 11.2

Aggiunta di una immagine

Per la creazione di un'immagine personalizzata seguire questa guida (https://zero-to-jupyterhub.readthedocs.io/en/latest/jupyterhub/customizing/user-environment.html#customize-an-existing-docker-image). Una volta che l'immagine è creata, caricarla sulla macchina e contattare il referente tecnico per la sua inclusione nella lista di immagini usate da jupyter (NB. l'immagine sarà visibile dagli altri utenti).

Utilizzo delle sole GPU prenotate.

Di default il notebook può accedere a tutte e 8 le GPU della macchina. Al fine di evitare conflitti l'utente è tenuto a caricare i modelli solo sulle GPU prenotate (gli strutturati che cedessero le credenziali di accesso a studenti, sono pregati di controllare che questa fondamentale regola sia **sempre** applicata).

Un possibile modo per farlo con il framework pytorch è usare questa sequenza di istruzioni, **prima di importare pytorch.**

```
import os
os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"] = "0,1,2"
os.environ["NVIDIA_VISIBLE_DEVICES"] = "0,1,2"
```

Queste istruzioni forzano pytorch a vedere soltanto il set di GPU 0,1,2.

Vi sono altri metodi utilizzabili per tensorflow e pytorch. Questi possono essere trovati ricercando opportunamente sui motori di ricerca. Non si fornisce supporto su questo aspetto, che è rimandato all'utente.

Utilizzo della macchina tramite SSH e Docker

Ogni utente può accedere al server con le proprie credenziali tramite il server ssh in ascolto sulla porta 22 all'indirizzo 192.168.60.200. Il sistema operativo è Ubuntu e sulla macchina è già configurato docker con il supporto per le GPU. Si ricorda che ogni utente deve eseguire i propri esperimenti in un container docker, in particolare l'utente **non deve installare pacchetti sulla macchina**.

L'utente può usare lo spazio nella sua home directory, non deve lasciare file al di fuori della home e non può lasciare dati sulla macchina in maniera stanziale (soprattutto se voluminosi).

L'utente può usare liberamente dei container da lui creati usando le immagini già presenti nella macchina, scaricati da internet o create da lui. Per permettere questo ogni utente ha accesso illimitato ai comandi di docker.

> Attenzione: L'accesso permette ad un utente di cancellare le immagini di altri utenti, oltre a permettere lo spegnimento (kill) di container attivi. **Pertanto, si raccomanda di prestare la massima attenzione quando si eseguono comandi che potrebbero potenzialmente impattare in modo catastrofico il lavoro degli altri utenti con cui la macchina è condivisa (es. cancellare immagini o spegnere containers).**

La persistenza dei dati va implementata montando dei volumi docker nei container generati che mappano su folders nella home utente (https://docs.docker.com/storage/volumes/).

Attenzione: container inattivi vengono periodicamente stoppati e eliminati dall'amministrazione, pertanto l'utente deve necessariamente salvare su un volume i dati importanti.

Esempio passo passo di uso dell'immagine di base con CUDA 11.2

Una volta nella macchina controllare di essere nella propria home

```
mrossi@dgxa100:~$ pwd
/raid/mrossi
```

Creare una folder per la persistenza dei dati

```
mrossi@dgxa100:~$ mkdir dati
mrossi@dgxa100:~$ ls
dati
```

Visualizzare la lista delle immagini disponibili sulla DGX.

```
mrossi@dgxa100:~$ docker image ls
REPOSITORY   TAG                              IMAGE ID     CREATED      SIZE
nvidia/cuda  11.4.0-devel-ubuntu20.04         b2c4c623c230 6 days ago   4.7GB
```

```
gpu-jupyter  latest                          c355c98c22bf 13 days ago  21.8GB
jupyter/tensorflow-notebook     latest       cbfd81abe559 2 weeks ago  3.06GB
nvcr.io/nvidia/pytorch          21.06-py3    e857099aa87e 2 weeks ago  14.5GB
nvcr.io/nvidia/tensorflow  21.06-tf1-py3    096da168123c 2 weeks ago  13.7GB
cschranz/gpu-jupyter            latest       b687f847bf8a 4 weeks ago  15GB
nvidia/cuda  11.0.3-cudnn8-ubuntu20.04       cba668ccbad9 5 weeks ago  3.75GB
jupyterhub/singleuser    1.4                 00b6c0cfb3b5 8 weeks ago  685MB
cschranz/gpu-jupyter  v1.4_cuda-11.0_20.04  5faea5f3925b 2 months ago 15GB
registry     <none>                          1fd8e1b0bb7e 2 months ago 26.2MB
nvidia/cuda   11.1-base-ubuntu20.04          287475453634 6 months ago 124MB
nvcr.io/nvidia/cuda       11.0-base          2ec708416bb8 10 months ago 122MB
```

In questo caso scegliamo la versione devel-ubuntu20.04 con cuda, creiamo quindi un container a partire da questa immagine (per informazioni sul concetto di container: https://www.docker.com/resources/what-container). Il container avrà una directory nella home mappata sulla folder dati.

```
mrossi@dgxa100:~$ docker run -it --gpus all -v /raid/mrossi/dati:/home/dati
--entrypoint /bin/bash --name nome_cognome_CUDA 2ec708416bb8
```

Spiegazione del comando:

- *-v* specifica il mapping tra la folder */raid/mrossi/dati* e la directory /home/dati
- *--name nome_cognome_CUDA* si specifica il nome del container. La nomenclatura consigliata è quella di chiamare i container con il proprio nome per minimizzare conflitti.
- L'ultima stringa è l'*id* dell'immagine CUDA ottenuto con il comando image ls.
- --gpus all seleziona tutte le GPU della macchina, questo è possibile solo quando l'utente le ha prenotate tutte (vedere punto sotto).

Apparirà quindi una shell di root all'interno del container. Su questa shell l'utente è libero di installare qualsiasi pacchetto o libreria. Ed eseguire i propri esperimenti salvando i risultati nel volume persistente.

```
root@f68975f15d98:/#
```

Attenzione: l'utente si accorge quando è nella shell del container perché vedrà il suo nome mutato in root (o nell'utente di default del container).

Utilizzo delle sole GPU prenotate

E' possibile avviare un container che ha accesso alle sole GPU prenotate seguendo queste istruzioni: https://github.com/NVIDIA/nvidia-docker/wiki/Frequently-Asked-Questions#i-have-multiple-gpu-devices-how-can-i-isolate-them-between-my-containers