Présentation Jean Zay

IRIT, Toulouse
15 décembre 2021

INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

cnrs
GENCI
Hewlett Packard
Enterprise
Jean Zay

GENCI IDRIS Jean Zay

Travailler sur Jean Zay

(pause)

Retex Passage à l'échelle (multi noeud / multi GPU)

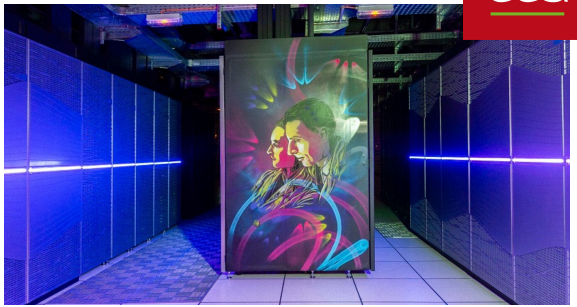
Grand équipement national de calcul intensif

Société civile coordinatrice des trois grands centres de calcul nationaux (2007)

GENCI
Le calcul intensif au service de la connaissance

Calculateur Jean Zay
IDRIS

Calculateur Joliot-Curie
TGCC



Calculateur Occigen (puis Adastra)
CINES

Grand équipement national de calcul intensif

Missions :

- promotion du calcul intensif pour la recherche académique et industrielle
- participer à l'écosystème HPC/IA au niveau européen
- achat des supercalculateurs
- distribution des ressources de calcul

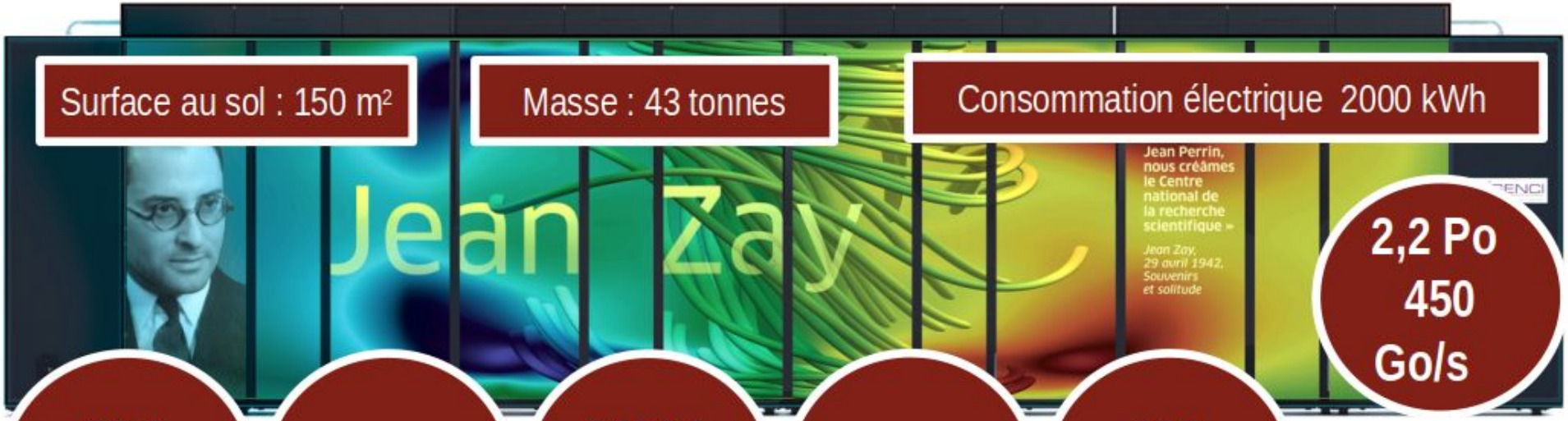
GENCI

Le calcul intensif au service de la connaissance

IDRIS : Centre national de calcul

INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

- Unité d'Appui à la Recherche créée en 1993
- Centre national Tier1 du CNRS pour le calcul numérique intensif de très haute performance (HPC) et l'intelligence artificielle (IA)
  - Hébergement et administration du supercalculateur Jean Zay
  - Centre d'excellence de support, conseil et expertise dans le domaine du HPC et de l'IA



Surface au sol : 150 m²

Masse : 43 tonnes

Consommation électrique 2000 kWh

28,2 Pflop/s

Puissance crête cumulée

2712

GPU NVIDIA V100 16-32 Go

86504 cœurs

Intel Cascade Lake 2.5 GHz

428 To

de RAM

100 Gb/s

Réseau d'interconnexion Omni-Path

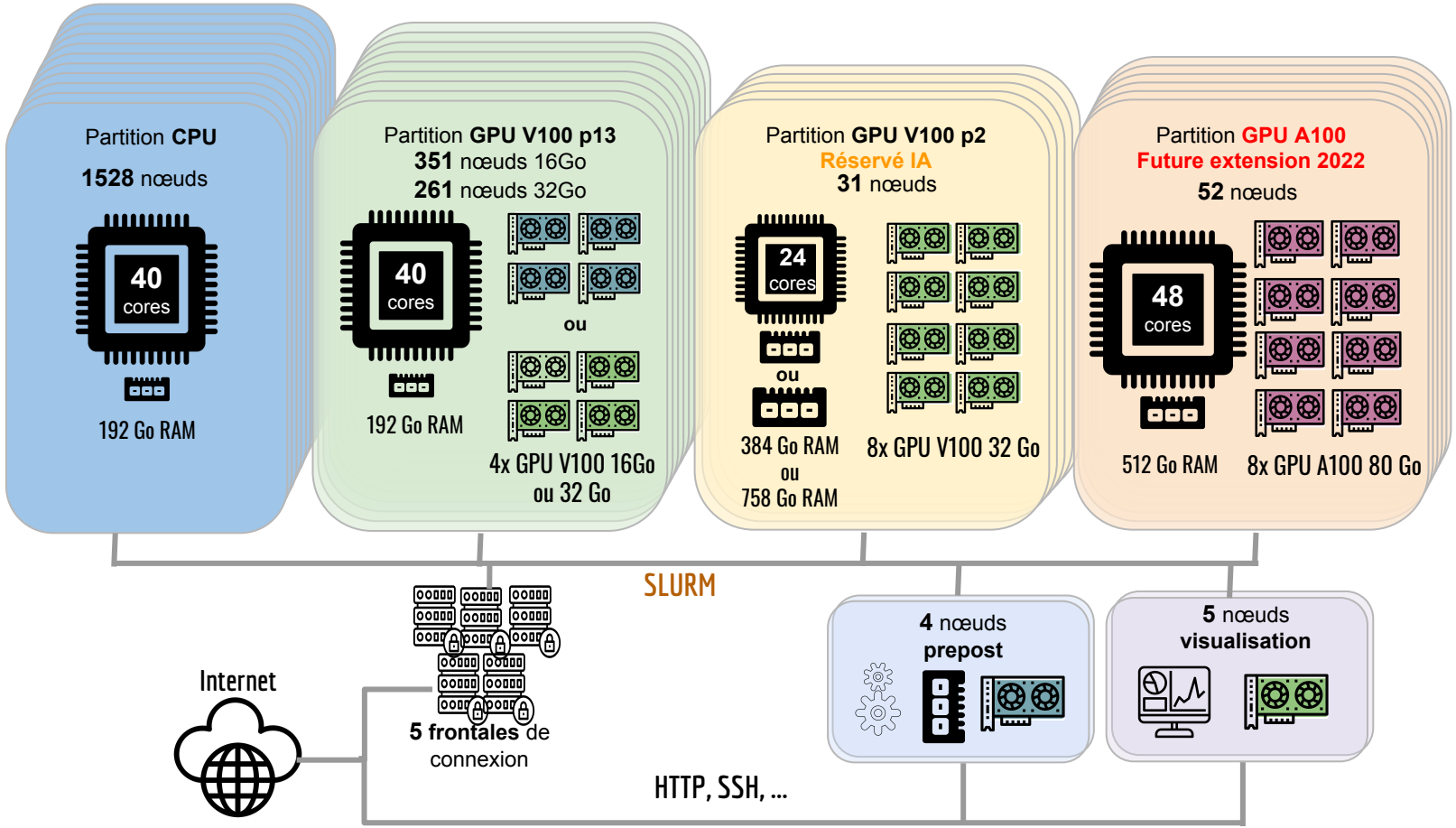
(1 lien par noeud scalaire et 4 liens par noeud convergé)

2,2 Po 450 Go/s

SSD

35 Po 150 Go/s

Disques mécaniques


Partition CPU
1528 nœuds
40 cores
192 Go RAM
Partition GPU V100 p13
351 nœuds 16Go
261 nœuds 32Go
40 cores
192 Go RAM
ou
4x GPU V100 16Go ou 32 Go
Partition GPU V100 p2
Réservé IA
31 nœuds
24 cores
ou
384 Go RAM ou 758 Go RAM
8x GPU V100 32 Go
Partition GPU A100
Future extension 2022
52 nœuds
48 cores
512 Go RAM
8x GPU A100 80 Go
SLURM
Internet
5 frontales de connexion
4 nœuds prepost
5 nœuds visualisation
HTTP, SSH, ...

Accès aux ressources

début 2022

AR : **Accès Régulier** pour les “gros” projets

- ≥ 500 000 heures CPU
- ≥ 50 000 heures GPU

Allocation pour une année

2 appels à projet par an

- Janvier - février :
  pour un démarrage en mai
- Juillet - septembre :
  pour un démarrage en novembre

AD : **Accès Dynamique** pour les autres projets

- < 500 000 heures CPU
- < 50 000 heures GPU

Allocation pour une année

Ouverts toute l’année

www.edari.com / GENCI

Compte utilisateur

- Un compte doit être rattaché à un projet
- La procédure dépend du centre de calcul
- IDRIS : Zone à Régime Restrictif

http://www.idris.fr/info/gestion/gestion-des-comptes.html / IDRIS

Les utilisateurs

11 comités thématiques HPC et IA

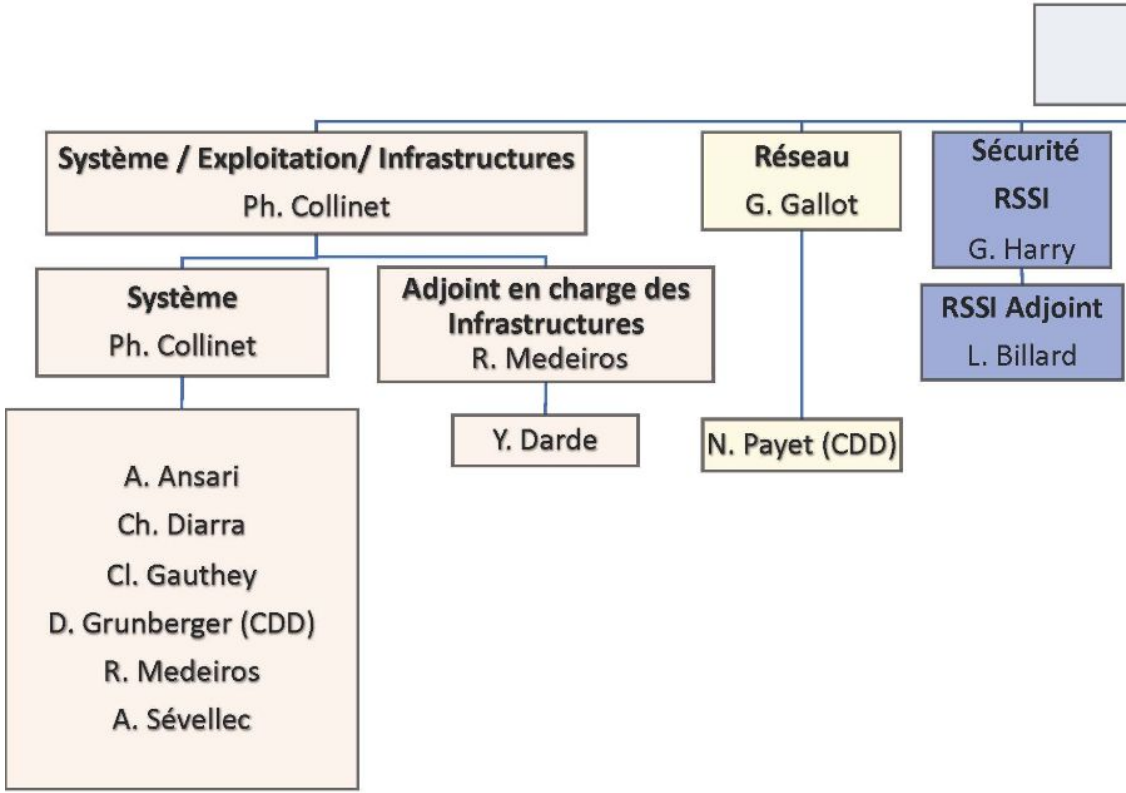
1 comité utilisateurs

> 2100 comptes actifs

CT1	: Environnement
CT2a	: Écoulements non réactifs
CT2b	: Écoulements réactifs et/ou multiphasiques
CT3	: Biologie et santé
CT4	: Astronomie et géophysique
CT5	: Physique théorique et physique des plasmas
CT6	: Informatique, algorithmique et mathématiques
CT7	: Dynamique moléculaire appliquée à la biologie
CT8	: Chimie quantique et modélisation moléculaire
CT9	: Physique, chimie et propriétés des matériaux
CT10	: Intelligence artificielle et applications transversales du calcul

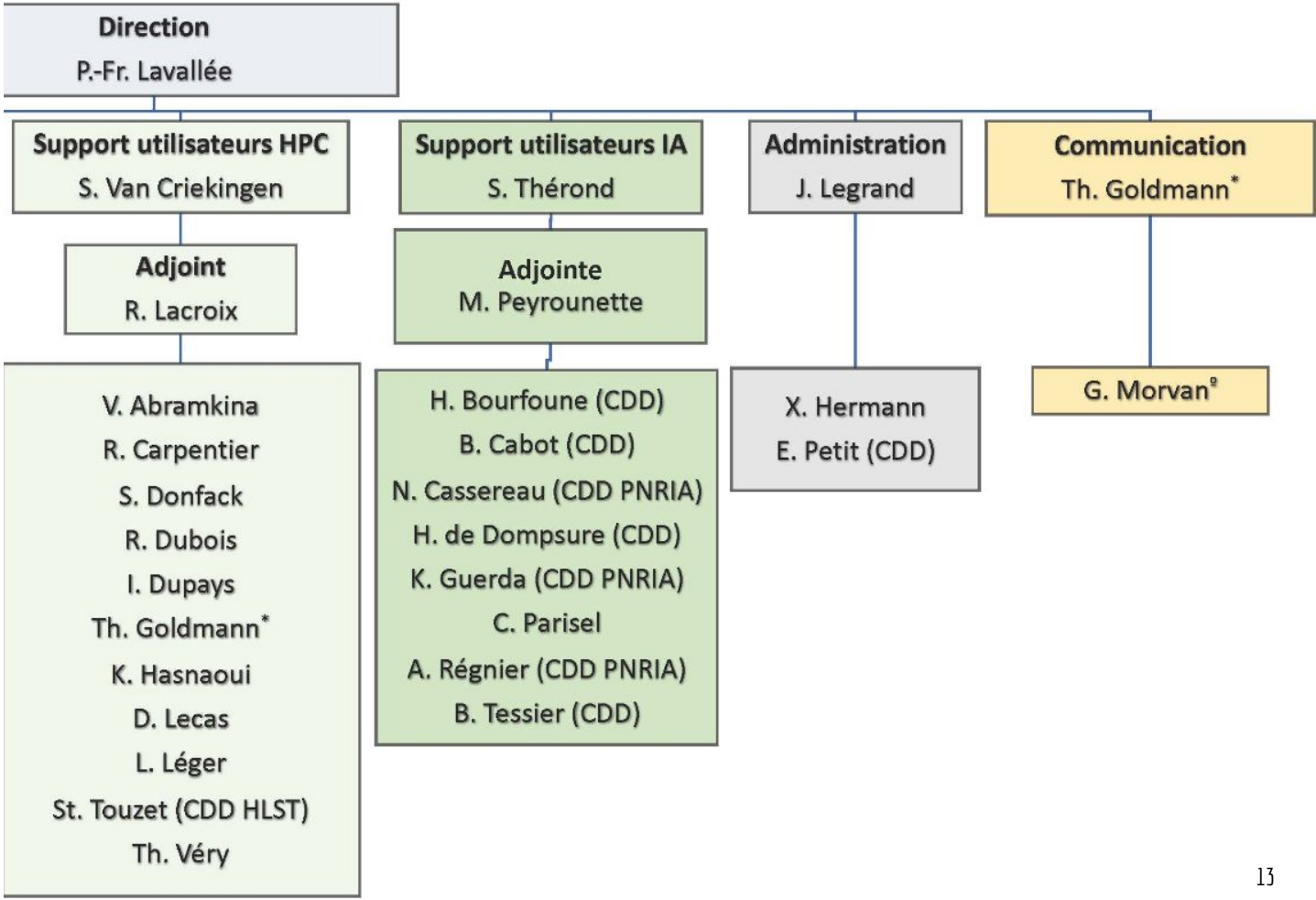
Personnel IDRIS

En charge de l'infrastructure



Personnel IDRIS

Au contact des utilisateurs



L'IDRIS, c'est aussi

- De la documentation en ligne : http://www.idris.fr/jean-zay/
- Des formations : http://www.idris.fr/formations/formations.html
- Des cours PRACE : http://www.idris.fr/prace/formationsptc.html
- Un support avancé : http://www.idris.fr/support_avance.html
- Des hackathons

Contacts et liens utiles

Demande d'heures de calcul :
- www.edari.fr
- www.genci.fr

Ouverture de compte de calcul à l'IDRIS :
- http://www.idris.fr/info/gestion/gestion-des-comptes.html

Site le l'IDRIS :
- www.idris.fr

Documentation Jean Zay :
- doc générale : www.idris.fr/jean-zay
- doc nouvel utilisateur : http://www.idris.fr/su/debutant.html
- doc IA : http://www.idris.fr/ia/index.html

Contacts :
- demande d'heures → acces@genci.fr
- ouverture de compte → gestutil@idris.fr
- utilisation des ressources → assist@idris.fr

cnrs
GENCI
Hewlett Packard
Enterprise
Jean Zay

GENCI IDRIS Jean Zay

Travailler sur Jean Zay

(pause)

Retex Passage à l'échelle (multi noeud / multi GPU)

Bien démarrer sur Jean Zay

Les bons réflexes

Se rappeler :

- La documentation (http://www.idris.fr/)
- Le support (assist@idris.fr)

Connexion

- À partir d’une machine déclarée (IP) : ssh ou putty

```
$ ssh login1@jean-zay.idris.fr
Last login: Sat Dec 11 19:29:31 2021 from 192.168.24.130
*****************************************************************
* Ceci est un serveur de calcul de l'IDRIS. Tout acces au systeme     *
* doit etre specifiquement autorise par l'IDRIS. Si vous tentez de    *
* de continuer a acceder cette machine alors que vous n'y etes pas    *
* autorise, vous vous exposez a des poursuites judiciaires.           *
*****************************************************************
* Orsay      CNRS / IDRIS - Frontale  -  jean-zay.idris.fr   France   *
*****************************************************************
*                 Maintenance hardware sur Jean Zay                   *
*             Le Mardi 14 Decembre 2021 de 8h30 a 12h00               *
*****************************************************************
[login1@jean-zay1: ~]$
```

Connexion

- À partir d’une machine déclarée
- À partir d’une machine de rebond

Fichier $HOME/.ssh/config :

```
host jean-zay jz
  hostname jean-zay.idris.fr
  user <your login>
  proxyjump <login_machine_de_rebond>@<machine_de_rebond>
```

Compte, projet et consommation

- Un **compte de calcul** par utilisateur
- … rattaché à un **projet** auquel des heures sont allouées
- Un projet peut avoir plusieurs utilisateurs

```
$ idracct
=======================================
 GPU (du 30-04-2021 au 26-04-2022)  Allocation : 1000 h.gpu
=======================================
  Compte   Proprietaire Consommation Nb travaux Remboursement
  -------- ------------ ------------ ---------- -------------
  login1   DUPONT Alex        152.00          5             -
  login2   DANTEC Pierre       87.00         12             -
  -------- ------------ ------------ ---------- -------------
                 Totaux       239.00         17          0.00
            Utilisation       23.90%
```

Compte, projet et consommation

- Un **compte de calcul** par utilisateur
- … rattaché à un **projet** auquel des heures sont allouées
- Un projet peut avoir plusieurs utilisateurs
- Un utilisateur peut appartenir à plusieurs projets

```
$ idrproj
Available projects
--------------------------
   abc (123456)
   def (127890) [default][active]
```

Les noeuds (1) : frontales ou noeuds de connexion

- Partagés par l'ensemble des utilisateurs
  - 1 coeur de calcul, et 30 minutes de temps CPU*
- Accès à des serveurs distants (git, wget)
- Pas de GPU

→Ne pas lancer de calculs longs ou lourds !

*: temps réel plus long

Les noeuds (2) : prepost et compil

- Pas de décompte d'heures
- Accès internet
- De 2h (defaut) à **20h**
- Accès via la directive `--partition`
- GPU disponible sur prepost (mais partagé !)

http://www.idris.fr/jean-zay/pre-post/jean-zay-pre-post_partition_slurm.html

Les noeuds (3) : noeuds de calcul

- Noeuds CPU
- Noeuds GPU
  - Par défaut : gpu_p13, avec des V100 de 16 ou 32 Go
  - #SBATCH -C v100-16g ou #SBATCH -C v100-32g

```
$ idrdoc
*****************************************************************************
*                        Jean Zay HPE SGI 8600 System                       *
*                               Configuration                               *
*****************************************************************************
Partitions / Quality of Service (QoS) / Storage disks
…
```

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-exec_partition_slurm.html

Espaces disques et quotas

Espaces **personnels**

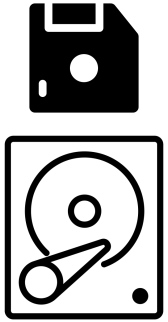
$HOME

3 Go

$WORK

5 To

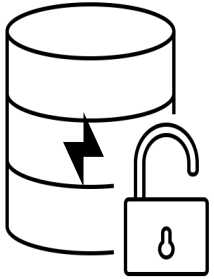
$STORE



50 To

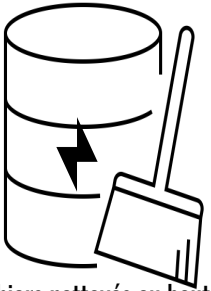
Espaces **partagés**

$DSDIR



Datasets publiques
dépôt sur demande

$SCRATCH



Fichiers nettoyés au bout de
30 jours sans activité

```
$ idrenv
export ALL_CCFRSCRATCH=[redacted]
export ALL_CCFRSTORE=[redacted]
export ALL_CCFRWORK=[redacted]
export HOME=[redacted]
```

Espaces disques et quotas

```
$ idr_quota_user
…
abc WORK
INODE:   |███░░░░░░░░░░░----|  U: 364k/5500k  6.62%   G: 462k/5500k  84.15%
STORAGE:|░░░░░░░░----------|  U: 98.83GB/48.83TB 0.20%  G: 24.12TB/48.83TB 49.40%

$ idr_quota_project
PROJECT: abc SPACE: WORK
PROJECT USED INODE: 462803/550000 84.15%
PROJECT USED STORAGE: 24.12 TiB/48.83 TiB 49.40%
```

LOGIN	INODE ▽	INODE %	STORAGE	STORAGE %
login1	728039	13.24%	244.44 GiB	0.49%
login2	624709	11.36%	781.95 GiB	1.56%
login3	431103	7.84%	10.74 TiB	21.99%

…

Modules : outils et environnement de calcul

- Mise à disposition d’un catalogue d’outils :
  - Librairies : `cuda/11.2`
  - De compilateurs : `gcc/9.3.0`
  - D’environnements conda : `tensorflow-gpu/py3/2.7.0`

Modules vs environnements personnels

Quelques pistes de réflexion

Modules

```
$ module load pytorch-gpu/py3/1.10.0
Loading pytorch-gpu/py3/1.10.0
  Loading requirement: cuda/11.2 nccl/2.9.6-1-cuda cudnn/8.1.1.33-cuda
gcc/8.3.1 openmpi/4.1.1-cuda
    intel-mkl/2020.4 magma/2.5.4-cuda sox/14.4.2
```

- **Compilation sur mesure** des paquets emblématiques **:**
  - Pytorch, horovod, …
- **472** “paquets pip” et **606** “paquets conda” !
- Demande d’ajout de paquets → ✉ assist@idris.fr

Nouveau module à chaque nouvelle version (x.**y**.**z**) de Pytorch et Tensorflow

Modules - ajouts personnels

1. ⚠️ Lien `$HOME/.local` → `$WORK/.local`
2. `module load <module>`
3. `pip install --no-cache-dir --user <paquet>`
   - va installer dans .local *
   - peut “écraser” une version d’un paquet du module (↗ ou ↘)
4. Ou `python setup.py` à partir d’un dépôt git
5. Ou compilation et déploiement via pip

Attention aux effets de bords !

Pour en savoir plus → la doc !

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-python-env.html

Jupyter Notebooks

- Exécutable sur tous les noeuds (après `srun` + `module load`)
  - `idrlab --notebook-dir=$WORK`

```
(pytorch-gpu-1.10.0+py3.9.7) $ idrlab --notebook-dir=.
INFO 2021-12-12 23:07:59,016 Starting Jupyter server. Please wait...:
INFO 2021-12-12 23:07:59,211 --Launching Jupyter server. Please wait before
attempting to connect...
INFO 2021-12-12 23:08:04,328 --Send data.
INFO 2021-12-12 23:08:04,329 Running command: idrjup-send --server -n 8000
-l0
INFO 2021-12-12 23:08:07,449 --Jupyter server launched. Please connect.
URL de connexion :     https://idrvprox.idris.fr
Mot de passe URL :     <mot de passe utilisateur>
Mot de passe jupyter : Fq8kYixZuToaSS0tDtjzyto6rcLCRe9J
```

http://www.idris.fr/jean-zay/pre-post/jean-zay-jupyter-notebook.html

Lancer des calculs

- Script slurm
- Commandes

Démarrer des jobs - script slurm

```
#!/bin/bash
#SBATCH --job-name=training
#SBATCH --output=log/%x_%j.out
#SBATCH --error=log/%x_%j.err
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=8
#SBATCH --gres=gpu:8
#SBATCH --cpus-per-task=6
#SBATCH --partition=gpu_p4
#SBATCH --hint=nomultithread

#SBATCH --time=00:30:00
#SBATCH --qos=qos_gpu-dev

#SBATCH --account=abc@gpu
```

Démarrer des jobs

```
#!/bin/bash
#SBATCH --job-name=training
#SBATCH --output=log/%x_%j.out
#SBATCH --error=log/%x_%j.err
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=8
#SBATCH --gres=gpu:8
#SBATCH --cpus-per-task=6
#SBATCH --partition=gpu_p4
#SBATCH --hint=nomultithread

#SBATCH --time=00:30:00
#SBATCH --qos=qos_gpu-dev

#SBATCH --account=abc@gpu
```

```
## load Pytorch module
module purge
module load pytorch-gpu/py3/1.8.0

export TMPDIR=$SCRATCH/tmp
## launch script on every node
set -x
time srun python -u demo.py -b 256
date
```

```
$ sbatch training.slurm
```

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-exec_partition_slurm.html
http://www.idris.fr/jean-zay/cpu/jean-zay-cpu-exec_jobarray.html

Reserver des ressources

- `sbatch` et `salloc` : allocation des ressources
- `srun` : lance les tâches sur les ressources allouées

```
$ srun -p compil -t 01:00:00 -c 10 --hint=nomultithread --pty bash
```

- Suivi

```
$ squeue -u $USER
224424    gpu_p2     runA   login1  R   11:59:29      4 jean-zay-ia[808-811]
226606    gpu_p2l    runB   login1 PD       0:00      2 (Resources)
1517697   prepost    clean  login1 PD       0:00      1 (Dependency)
```

- Annulation : `scancel $JOBID`
- Information sur un job : `scontrol show job $JOBID`

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-exec_controle_batch.html

cnrs
GENCI
Hewlett Packard
Enterprise
Jean Zay

GENCI IDRIS Jean Zay

Premiers pas sur Jean Zay

(pause)

Retex Passage à l'échelle (multi noeud / multi GPU)

Etapes

- Préparatifs
- Adaptations
- Exemples de mise en oeuvre

Deux publics

IA = **outil** à appliquer sur un domaine d'application

- Expertise sur les données, les problématiques

IA = **sujet de recherche**

- Expertise sur le Deep Learning
- Benchmarks via des datasets de référence

Deux publics

IA = **outils** à appliquer sur un domaine d'application

- Expertise sur les données, les problématiques

IA = **sujet de recherche**

- Expertise sur le Deep Learning
- Benchmarks via des datasets de référence

Focus sur :

- Trouver le bon outil (fiable, stable, adaptable)
- Optimiser (usage GPU, choix stockage)
- Distribuer sur plusieurs noeuds et GPU
- Exploiter au mieux l'infrastructure Jean Zay

$\nexists$ recettes magiques !

Objectifs et Préparatifs

- Exploiter de plus **gros modèles,** paralléliser

Attentes :

- Faire **converger** le modèle
- Avoir une **durée d'entraînement** optimale

Objectifs et Préparatifs

- Exploiter de plus **gros modèles,** paralléliser

Attentes :

- Faire **converger** le modèle
- Avoir une **durée d'entraînement** optimale

Première étape, avoir une référence

- Utiliser au mieux la mémoire d'un GPU (batch, *mixed precision*)
- Avoir des métriques
  - Temps de calculs
  - Graphes de convergence (Loss, Accuracy, ...)
  - Occupation mémoire GPU
- Gérer la reproductibilité (doc pytorch)

http://www.idris.fr/jean-zay/pre-post/jean-zay-profile-tensorflow-pytorch.html

Objectifs et Préparatifs

- Exploiter de plus **gros modèles,** paralléliser

Ne pas hésiter à utiliser :

- Les **loggers** (→tensorboard, wandb offline, mlflow)
- Les **profilers** (→pytorch, nsight)

Première étape, avoir une référence

- Utiliser au mieux la mémoire d'un GPU (batch, *mixed precision*)
- Avoir des métriques
  - Temps de calculs
  - Graphes de convergence (Loss, Accuracy, ...)
  - Occupation mémoire GPU
- Gérer la reproductibilité (doc pytorch)

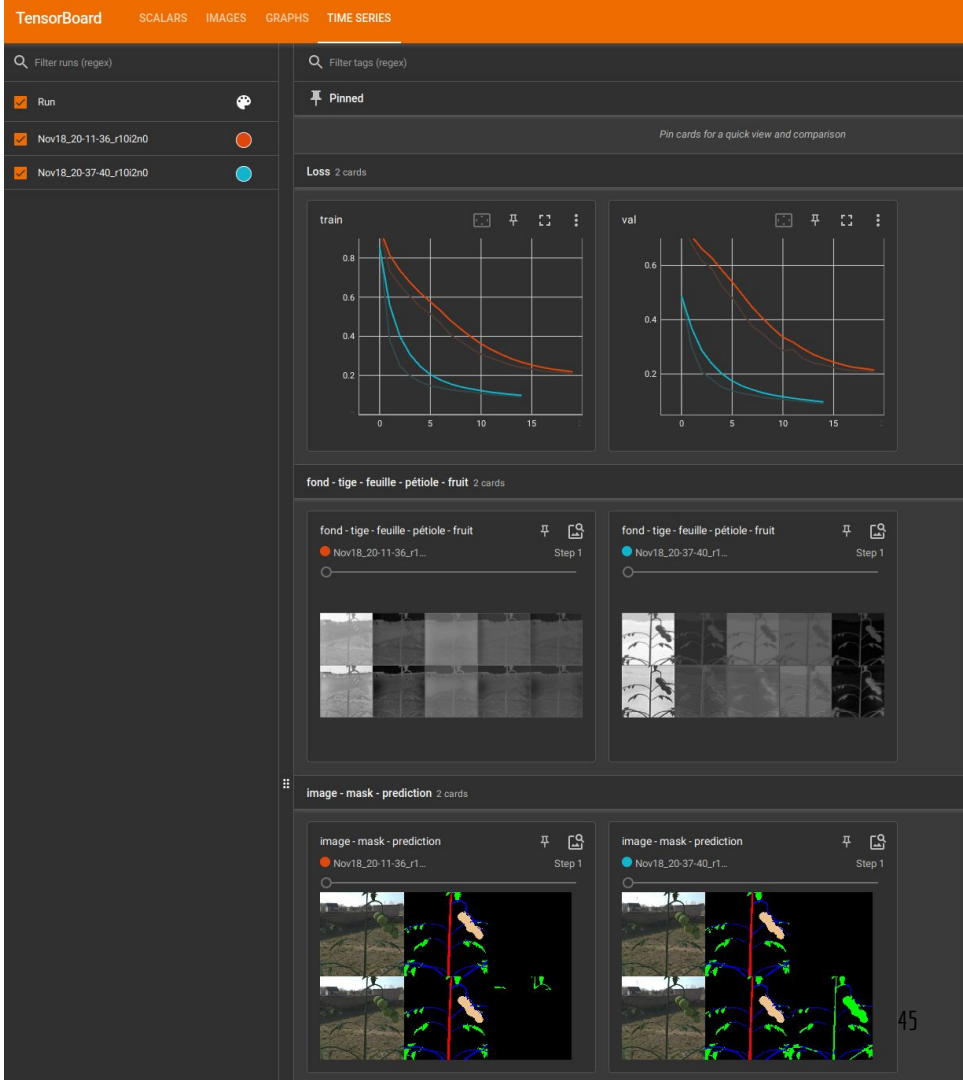
http://www.idris.fr/jean-zay/pre-post/jean-zay-profile-tensorflow-pytorch.html

Tensorboard

- Disponible dans nos modules pytorch et tensorflow, via **jupyter**
- Facile à utiliser dans les codes pytorch, lightning et tensorflow

Alternative :

- Wandb en mode **offline**
- MLFlow



Profilers

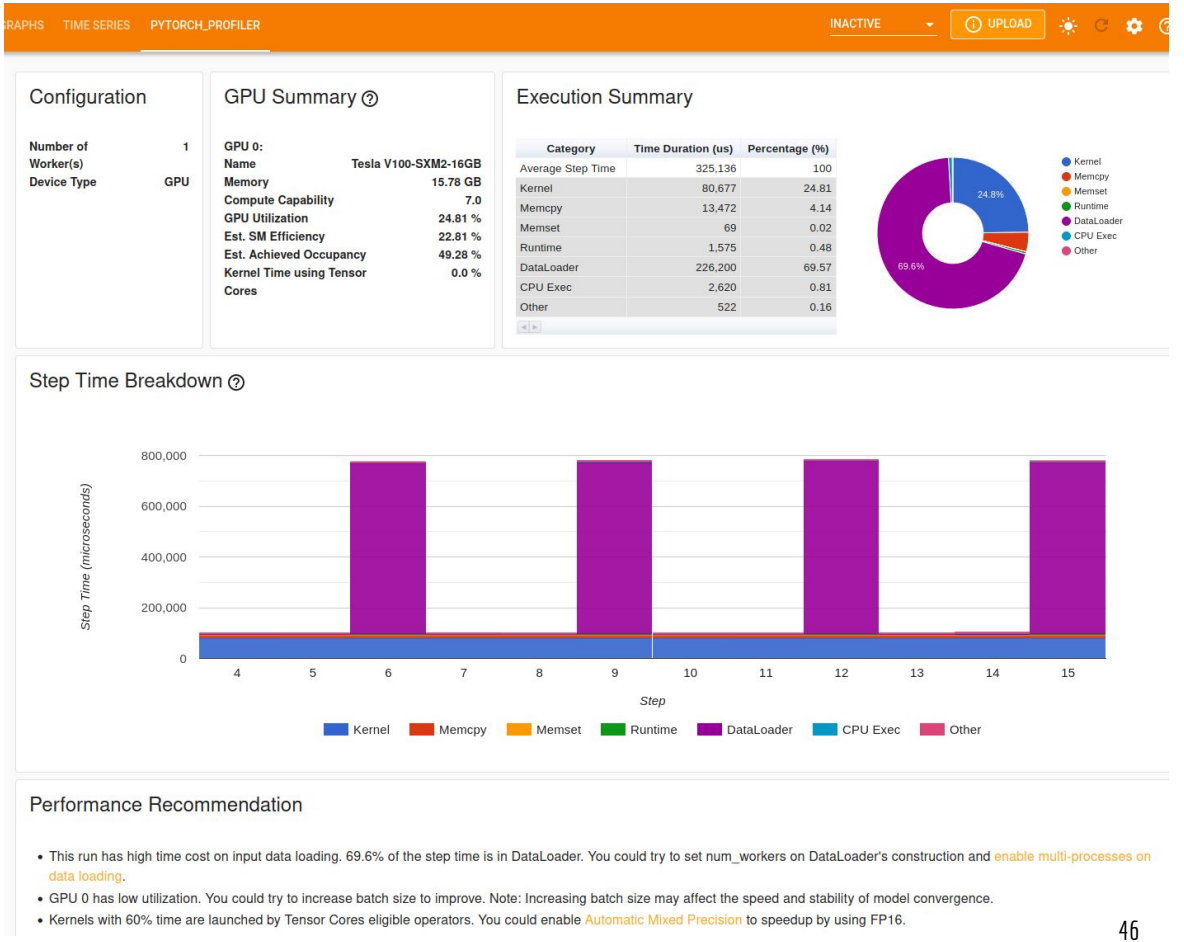
⚠️Visualiseurs non disponibles sur JZ

Deux choix (à partir de pytorch 1.8.1) :

- DLprof (+ nsight) de NVIDIA
- Pytorch profiler

Fonctionne en mode distribué :

- Attention à la **taille des logs** !



Impact du hardware

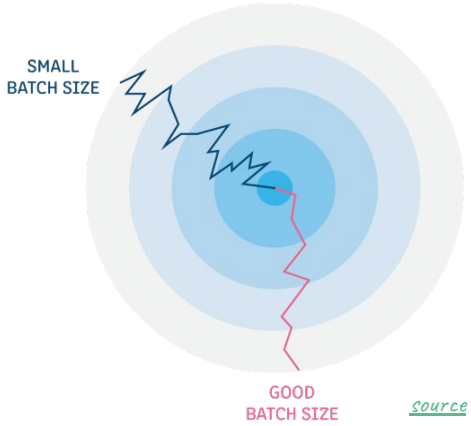
- Communications inter et intra noeuds : NCCL (GPU), MPI (CPU)
- Exploiter la *mixed precision*
- Accès disque : préférer `$SCRATCH` (SSD) ou `$DSDIR` (contient les datasets)
- Choix du GPU, QoS, partition :
  - Pas les mêmes disponibilités !
- Attention à la comptabilité !
  - Pour un job multi noeuds, 100% des GPU d'un noeud sont comptabilisés

Adaptations

- Taille du batch et learning rate
  - Petit → apprentissage lent
  - Grand → convergence du modèle difficile

Proposition (Horovod)

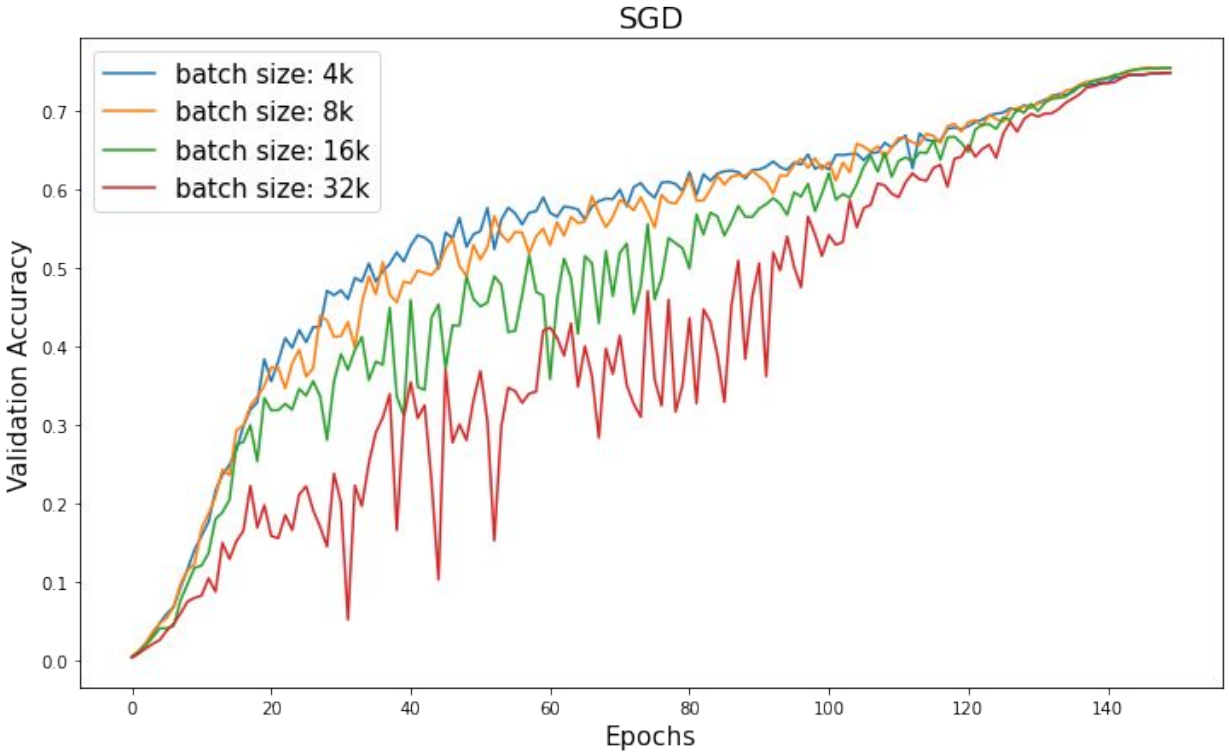
- `Batchsize x k → Learning rate x k`
- Augmenter la taille du batch progressivement durant un entraînement



source

Adaptations code

- Taille du batch
- Learning Rate (LR)



Adaptations code

- Taille du batch
- Learning Rate (LR)
- Scheduler
  - Trouver la valeur initiale de LR
  - Trouver la stratégie d’évolution
  - Ajouter une période de *warmup* (LR faible sur les premières époques)

Adaptations code

- Taille du batch
- Learning Rate (LR)
- Scheduler
- Optimizer
  - Spécifique pour des grands batchs (voir librairie NVIDIA apex)

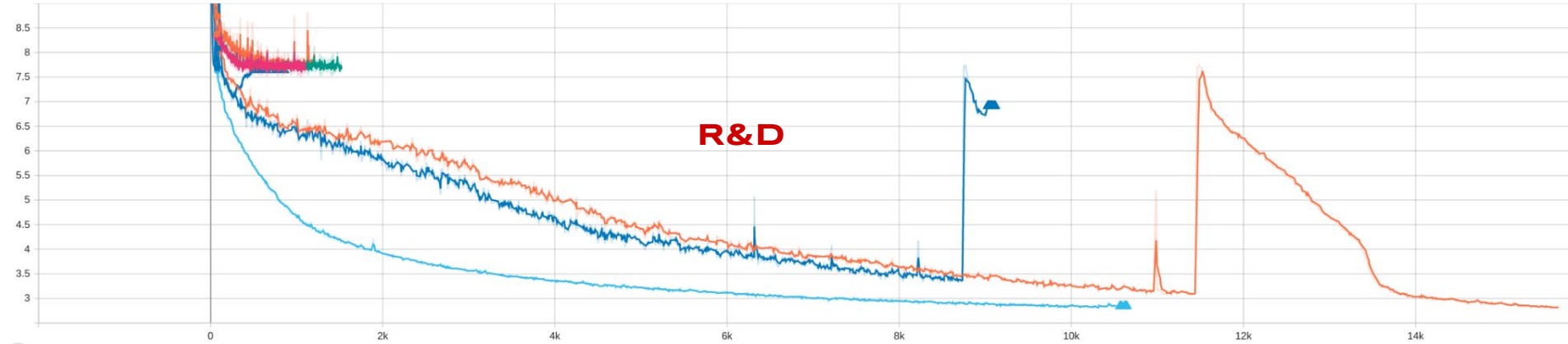
Big Science (modèle 104B) - 15 métriques suivies

TensorBoard SCALARS TEXT TIME SERIES

PREVIOUS PAGE NEXT PAGE

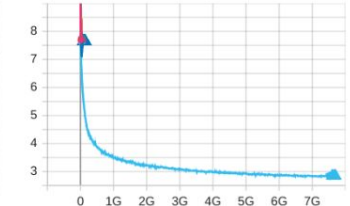
Pour plus d'information : chroniques

lm-loss-training/lm loss
tag: lm-loss-training/lm loss

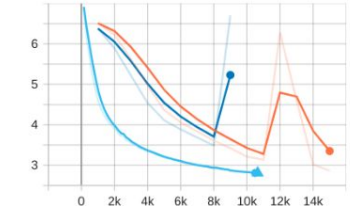


Name	Smoothed	Value	Step	Time	Relative
tensorboard/base-exp-11	8.37	8.315	72	Mon Oct 18, 02:55:28	9m 2s
tensorboard/base-exp-12	8.083	7.971	78	Fri Nov 5, 03:59:53	14m 5s
tensorboard/bnb-exp-01	8.301	8.286	78	Wed Nov 3, 06:26:37	23m 15s
tensorboard/bnb-exp-02	8.326	8.314	78	Thu Nov 4, 08:51:46	23m 9s
tensorboard/bnb-exp-03	8.334	8.317	78	Thu Nov 4, 16:32:43	22m 28s
tensorboard/bnb-exp-04	8.737	8.738	78	Thu Nov 4, 21:29:08	22m 30s
tensorboard/cl-exp-01	7.686	7.657	78	Fri Oct 22, 21:41:08	1h 56m 15s
tensorboard/cl-exp-02	7.633	7.471	78	Sun Oct 24, 14:10:42	2h 21m 25s

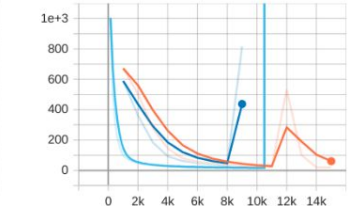
lm-loss-training/lm loss vs tokens
tag: lm-loss-training/lm loss vs tokens



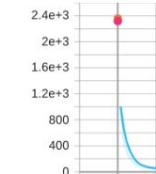
lm-loss-validation/lm loss validation
tag: lm-loss-validation/lm loss validation



lm-loss-validation/lm loss validation ppl
tag: lm-loss-validation/lm loss validation ppl



lm-loss-validation/lm los
tag: lm-loss-validation/lm lo
(without embeddings)



Passage à l'échelle

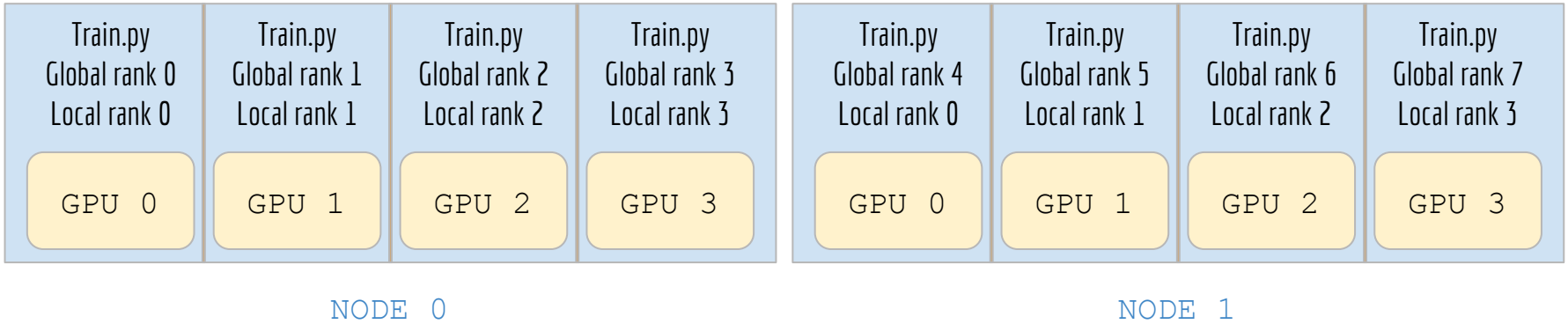
- Pytorch
- Launcher pytorch + Megatron LM
- Deepspeed

Pytorch

Utilisation de Distributed Data-Parallel (DDP)

Principes

- Code python lancé via `srun`
- Dans le code pytorch, on gère la mise en place des communications grâce aux variables d'environnement SLURM



https://github.com/pytorch/examples/blob/master/distributed/ddp/README.md

Slurm

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4
#SBATCH --cpus-per-task=10

module load pytorch-gpu/py3/1.10.0
 …

export NCCL_DEBUG=INFO
export NCCL_DEBUG_SUBSYS=ALL

srun python train.py
```

Python - récupération des variables SLURM

```python
import torch.distributed as dist
from torch.nn.parallel import DistributedDataParallel

if __name__ == '__main__':
    # get distributed configuration from Slurm environment
    NODE_ID = int(os.environ['SLURM_NODEID'])
    NTASKS_PER_NODE = int(os.environ['SLURM_NTASKS_PER_NODE'])
    hostnames = hostlist.expand_hostlist(os.environ['SLURM_JOB_NODELIST'])
    MASTER_ADDR = hostnames[0]

    rank, local_rank = int(os.environ['SLURM_PROCID']), int(os.environ['SLURM_LOCALID'])
    size = int(os.environ['SLURM_NTASKS'])
    cpus_per_task = int(os.environ['SLURM_CPUS_PER_TASK'])
```

Ou “`import idr_torch`”

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-torch-multi.html

Python - setup distribution

```
torch.cuda.set_device(local_rank)
dist.init_process_group(backend="nccl", init_method='env://',
                        world_size=size, rank=rank)
```

Python - setup dataset et model

```
from torch.utils.data.distributed import DistributedSampler
from torch.nn.parallel import DistributedDataParallel

ddp_model = DistributedDataParallel(model,device_ids=[local_rank])

batch_size_per_gpu = batch_size // size
train_sampler = DistributedSampler(train_ds, num_replicas=size,
                                   rank=rank)
train_loader = DataLoader(dataset=train_ds,
                          batch_size=batch_size_per_gpu,
                          sampler=train_sampler)
```

Python - exécution sur master / all

```
# calcul de Loss, métriques pour log
acc= ...
dist.all_reduce(acc, op=dist.ReduceOp.SUM)

# calcul à ne faire que sur le noeud master
if rank == 0:
    ...

# synchronisation
dist.barrier()
```

Alternative : Horovod

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-hvd-tf-multi.html

Pytorch launcher + Megatron LM

Lancement des jobs via Torch.distributed.launch
et parallélisation du modèle via Megatron LM

NVIDIA / Megatron-LM (git)

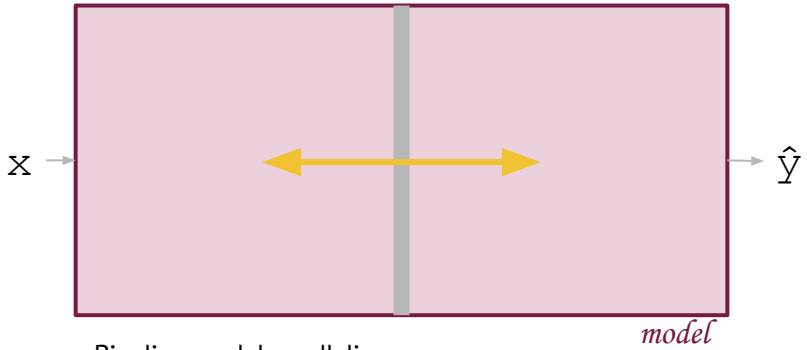
- Outil spécialisé dans l'entraînement de modèles **transformer** de **(plusieurs millions ou) milliards de paramètres**
- Propose des *checkpoints* de modèles (GPT, BERT, T5)
- Benchmarks GLUE implémentés et possibilité d'adapter à d'autres tâches*
- Propose une liste de *tokenizer* et vocabulaire
- Compatible avec Deepspeed (utilisé sur le projet BigScience)

Etat de l'art en NLP !

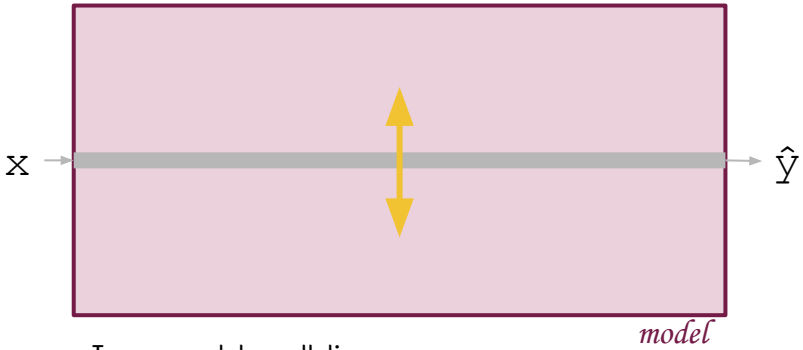
(*) : à condition d'étendre le code source

Principes

- Gestion de modèles volumineux



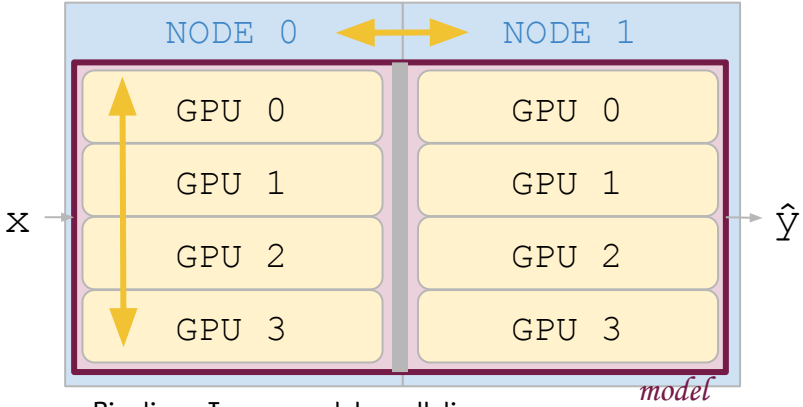
Pipeline model parallelism



Tensor model parallelism

Principes

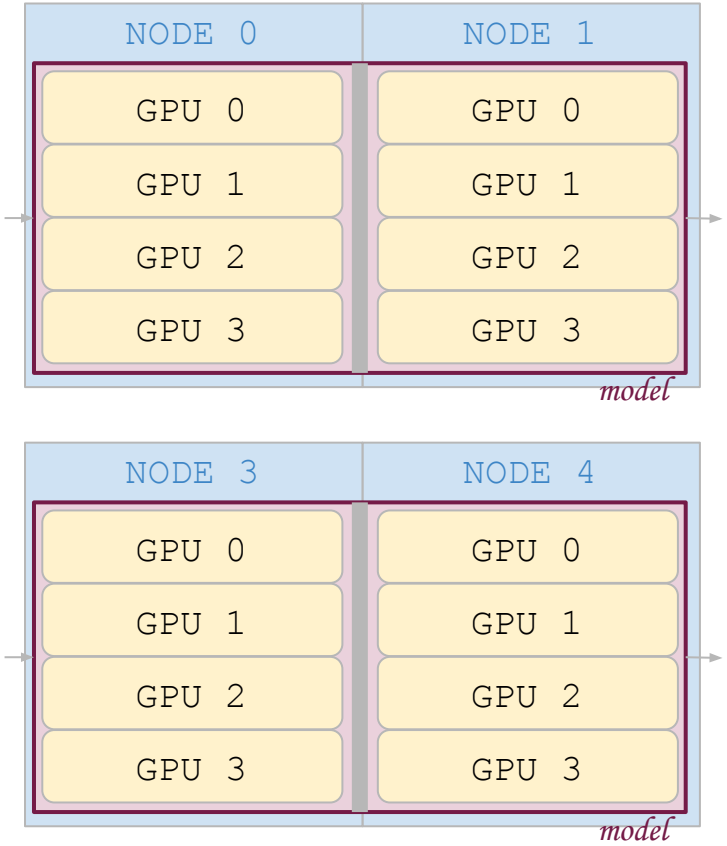
- Gestion des modèles volumineux



Pipeline + Tensor model parallelism

Démo

- bert-large (> 350M de paramètres)
- 1 modèle est réparti sur :
  - **2 noeuds x 4 GPU**
- Le parallélisme de données est sur **deux instances du modèle**



→ Toute la configuration se fait dans le script slurm

Slurm - setup pour megatron

```
export MEGATRON="./Megatron-LM/tasks/main.py  \
--task MNLI --train-data … --valid-data … \
--tokenizer-type BertWordPieceLowerCase \
--vocab-file bert-large-uncased-vocab.txt \
--epochs 10 \
--tensor-model-parallel-size 4 \
--pipeline-model-parallel-size 2 \
--num-layers 24 --hidden-size 1024 --num-attention-heads 16 \
--micro-batch-size 8 \
--checkpoint-activations \
--lr 5.0e-5 --lr-decay-style linear --lr-warmup-fraction 0.065 \
--seq-length 512 --max-position-embeddings 512 \
--log-interval 1 \
--weight-decay 1.0e-1 \
--distributed-backend nccl \
--fp16"
```

Slurm - setup du launcher

```
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=1
#SBATCH --gres=gpu:4
…

NNODES=$SLURM_JOB_NUM_NODES
MASTER_ADDR=$(hostname)
MASTER_PORT=$((10000 + $RANDOM % 1000))

export LAUNCHER="python -u  -m torch.distributed.launch  --nproc_per_node 4  \
--nnodes $NNODES \
--master_addr $MASTER_ADDR \
--master_port $MASTER_PORT"

srun bash -c '$LAUNCHER --node_rank $SLURM_PROCID  $MEGATRON'
```

Extrait fichier “output” du job

```
+ export 'LAUNCHER=python -u -m torch.distributed.launch --nproc_per_node 4
--nnodes 4 --master_addr r12i5n7 --master_port 6448'
+ LAUNCHER='python -u -m torch.distributed.launch --nproc_per_node 4
--nnodes 4 --master_addr r12i5n7 --master_port 6448'
+ srun bash -c '$LAUNCHER --node_rank $SLURM_PROCID $MEGATRON'

using world size: 16, data-parallel-size: 2, tensor-model-parallel size: 4,
pipeline-model-parallel size: 2
setting global batch size to 16

…

> initializing torch distributed ...
```

Deepspeed

Lancement de job avec Deepspeed

Microsoft / Deepspeed (site)

- Outil spécialisé dans l'**optimisation des ressources** GPU et des **communications** GPU/CPU/SSD
- Intégré dans transformer (Hugging Face) et Pytorch Lightning
- Librairie **pas** installée dans les modules IDRIS

Points forts :

- Zero Redundancy Optimizer (ZeRO) : ↗ taille des modèles (x10)
- 1-bit Adam and 1-bit LAMB optimizer : ↘ volume de communications (x5)

Installation

```
$ module load pytorch-gpu/py3/1.10.0
$ pip install --no-cache-dir --user  ninja
$ DS_BUILD_OPS=1 <DS*=1> DS_BUILD_AIO=0 pip install --no-cache-dir --user
deepspeed
$ $HOME/.local/bin/ds_report
--------------------------------------------------
DeepSpeed C++/CUDA extension op report
op name ................ installed .. compatible
--------------------------------------------------
cpu_adam ............... [YES] ...... [OKAY]
sparse_attn ............ [YES] ...... [OKAY]
transformer ............ [YES] ...... [OKAY]
…
async_io ............... [NO] ....... [NO]
--------------------------------------------------
deepspeed wheel compiled w. ...... torch 1.10, cuda 11.2
```

Fichier json

- Configuration deepspeed
  - Gradient accumulation : divise les batches en mini-batches séquentiels
  - Mixed precision : diminue l'occupation GPU du modèle
  - ZeRO 1 : premier niveau d'optimisation (sur 3)

```
{ "train_micro_batch_size_per_gpu": 32,
    "gradient_accumulation": 1,
    "fp16": {
      "enabled": true,
      "loss_scale": 0,
      "initial_scale_power": 32,
      "loss_scale_window": 1000,
      "hysteresis": 2,
      "min_loss_scale": 1
    },
    "zero_optimization": {
          "stage":1,
          "reduce_bucket_size": 5e8
    }
  }
```

Slurm

```
#!/bin/bash
#SBATCH --job-name=resnetx16_torch
#SBATCH --gres=gpu:4
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=4
#SBATCH --hint=nomultithread
#SBATCH --time=00:30:00
#SBATCH --cpus-per-task=10
#SBATCH -C v100-32g

module purge
module load pytorch-gpu/py3/1.10.0

## launch script on every task
set -x
time srun python -u deepspeed-resnet101.py  --deepspeed_config ds_zero1.json
```

Code python

```
parser = argparse.ArgumentParser()
# Include DeepSpeed configuration arguments
ds_parser = deepspeed.add_config_arguments(parser)
args = parser.parse_args()

# Initialize DeepSpeed to use the following features
# 1) Distributed model
# 2) Distributed data loader
# 3) DeepSpeed optimizer
ds_model, optimizer, _, _ = deepspeed.initialize(args=args, model=model,
                                                 model_parameters=parameters, optimizer=optimizer)
```

http://www.idris.fr/jean-zay/gpu/jean-zay-gpu-hvd-tf-multi.html

Autres options

- (Facebook) Fairscale
- (Hugging Face) accelerate
- Pytorch Lightning
- ...

cnrs
GENCI
Hewlett Packard
Enterprise
Jean Zay