

L'entraînement de réseaux neuronaux, en pratique

Atelier pour IFT780, Hiver 2022



compute | **calcul**
canada | canada

Par : Carl Lemaire


Calcul Québec

Plan

1. Bibliothèques logicielles
2. Matériel
3. Gestion des données
4. Atelier

Les bibliothèques logicielles

Numpy, Scikit-Learn, TensorFlow, PyTorch

Bibliothèques logicielles

	Spécialité	Caractéristiques	Matériel
Numpy	Mathématiques numériques	Peut être utilisé pour faire des NN (don't)	CPU
Scikit-Learn	ML général	Variété de modèles encapsulés et standardisés (ex. NN)	CPU
PyTorch TensorFlow (TF)	Deep Learning	Flexibilité et expressivité pour les NN	CPU GPU TPU



Matériel

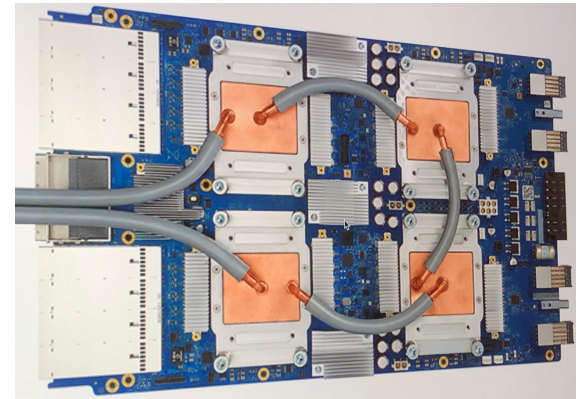
Accélérateurs

- GPU

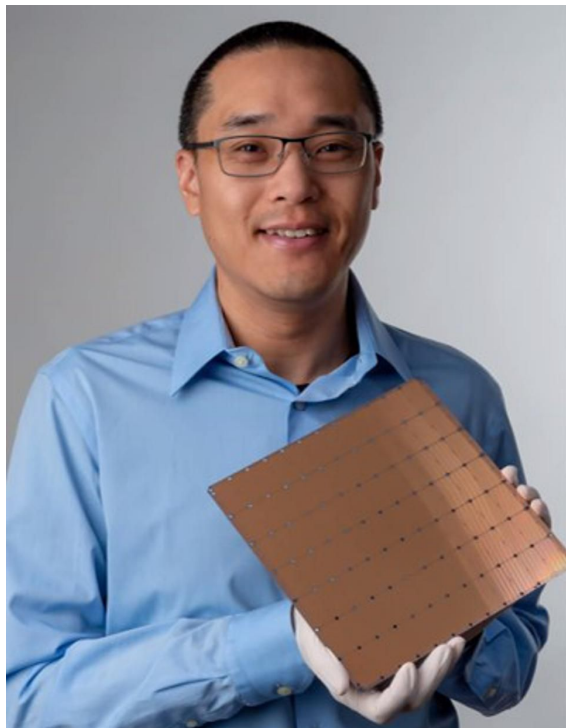
- **Graphics** Processing Unit
- Spécialisé pour: **Géométrie vectorielle**

- TPU

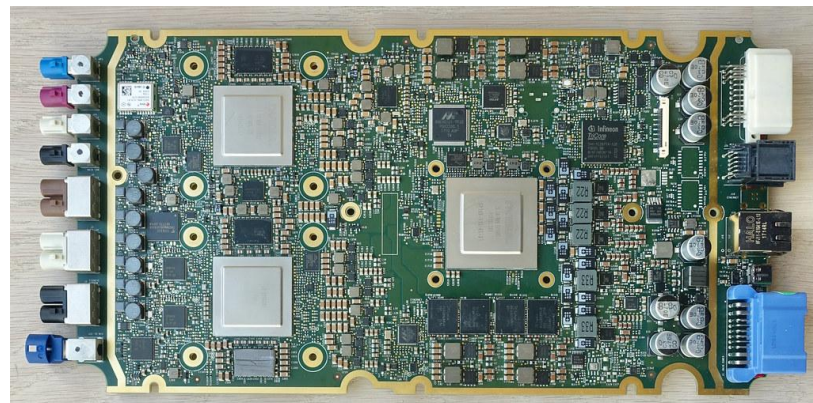
- **Tensor** Processing Unit
- Spécialisé pour: **Deep Learning**
- ~30x plus de performance/Watt en inférence v.s. GPU (Jouppi et al. 2017)
- Seulement chez Google Cloud (à l'origine)



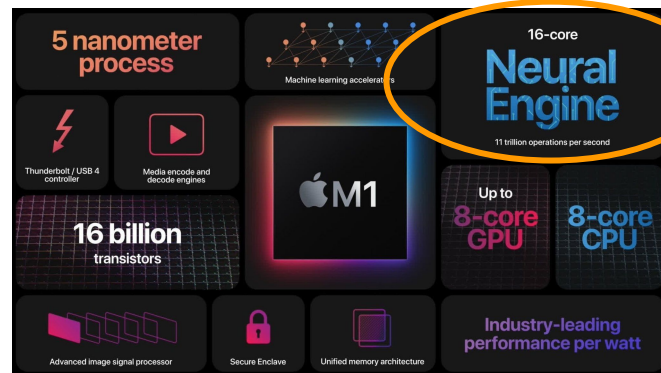
Accélérateurs



Cerebras



Tesla



Apple

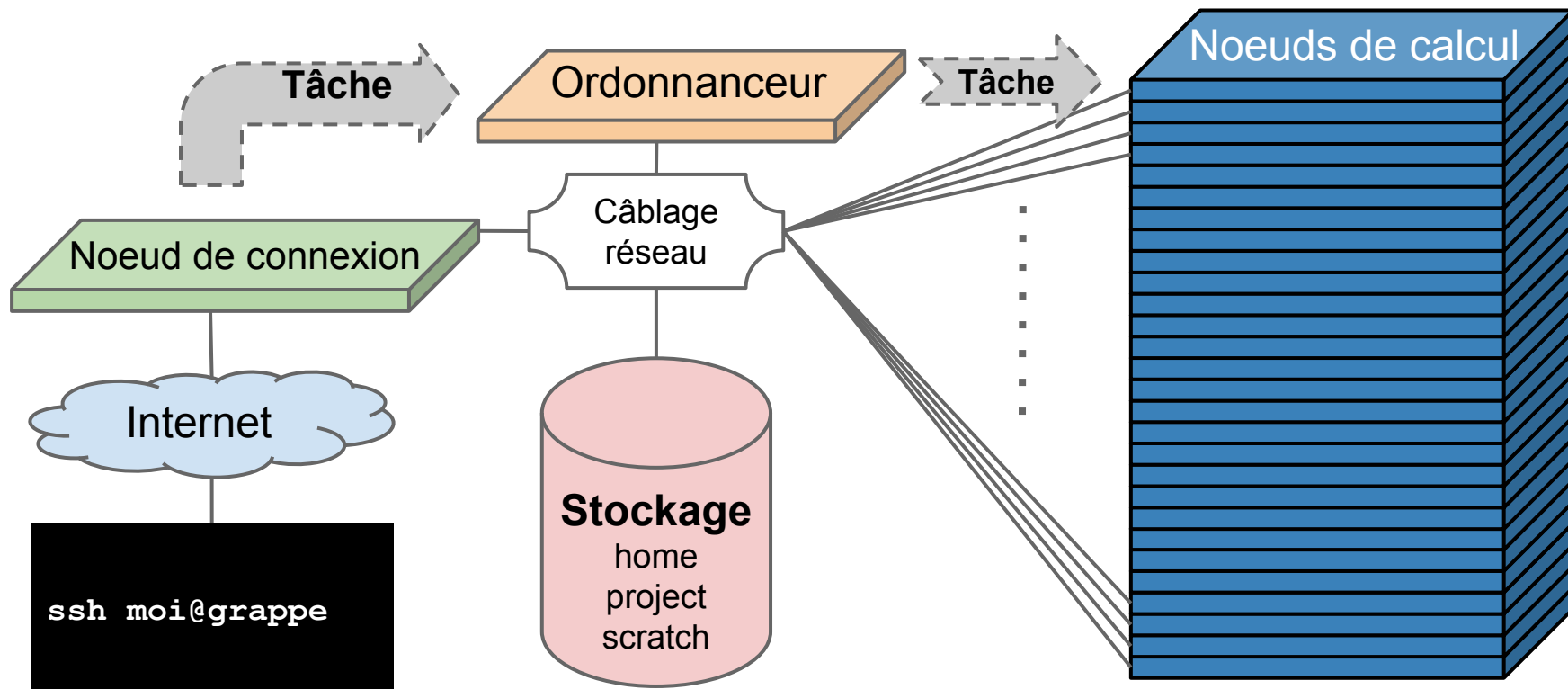
Grappe de calcul

a.k.a. “Superordinateur”

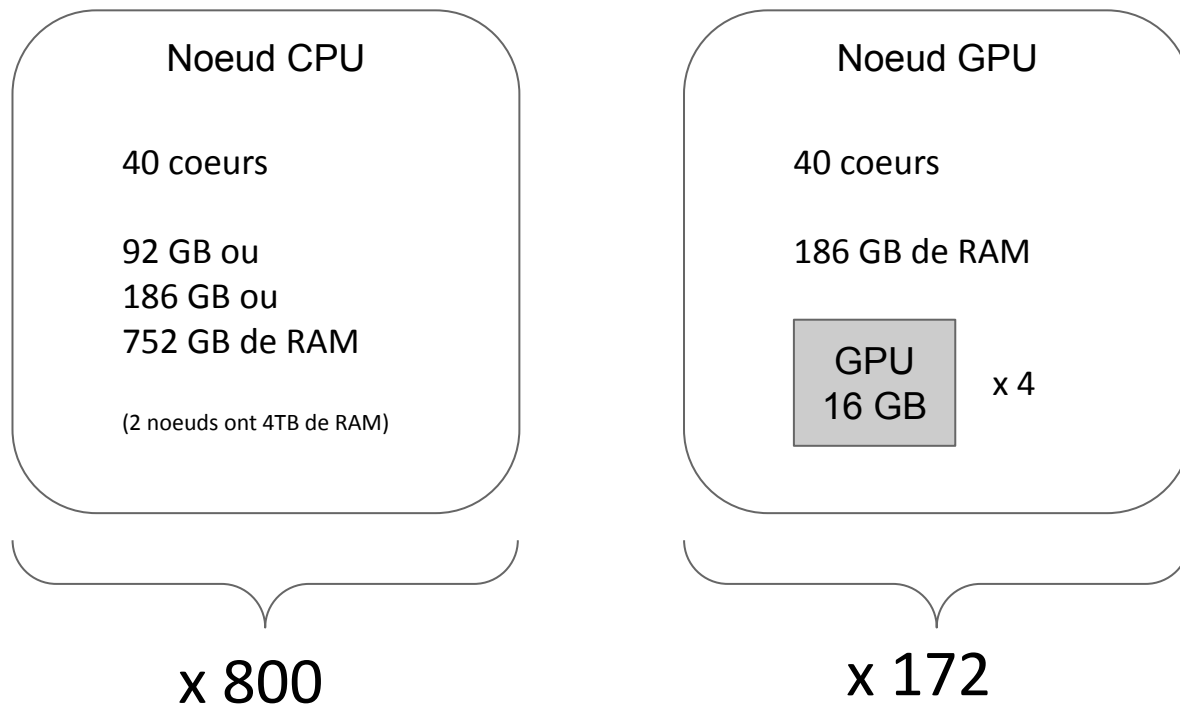
- Centaines de noeuds (ordinateurs)
- Connectés ensemble, et connectés à des noeuds spéciaux
- Système partagé par des centaines de personnes
- Stockage partagé (**Douleur partagée**)

À lire: [Using Compute Canada
Clusters for ML Research](#)

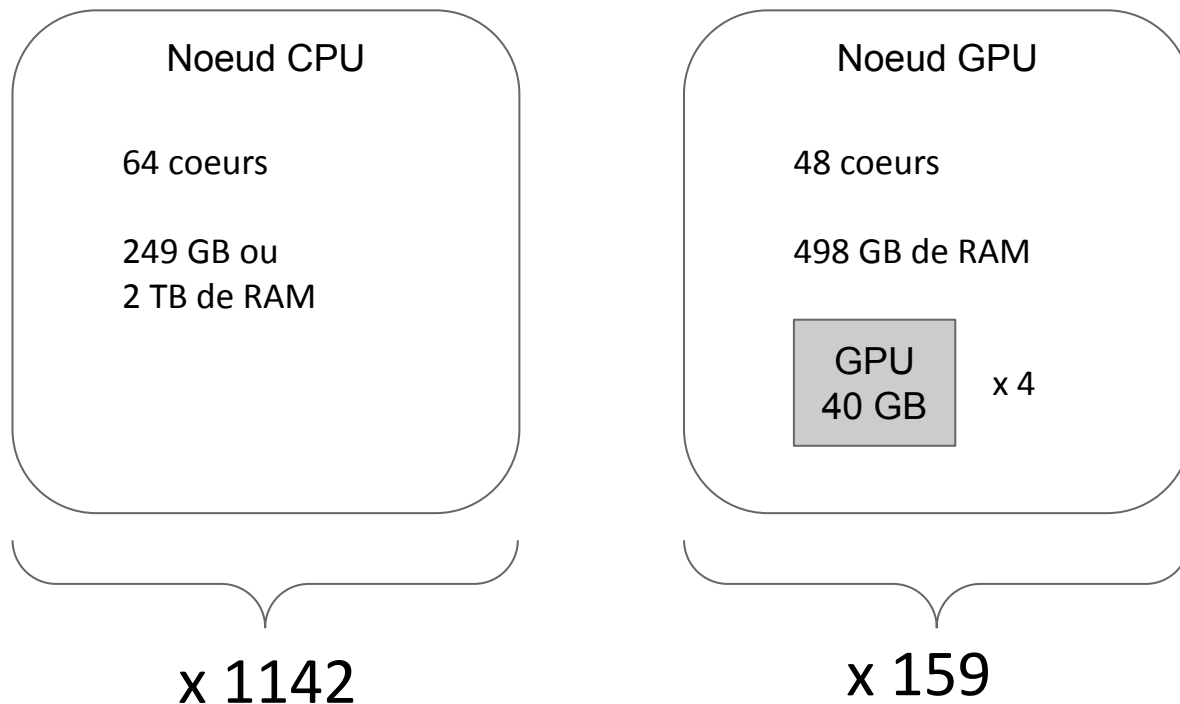
Grappe de calcul



La grappe Béluga



La grappe Narval



La juste part

- Le temps de calcul est partagé
- Le partage est fait selon un principe de juste part
- File d'attente avec priorité
- Priorité en fonction de l'utilisation récente
 - Si vous avez peu calculé dernièrement, vous aurez une meilleure priorité

Soumission de tâches

- Vous soumettez une tâche, qui sera ajoutée à une file d'attente
- Votre script sera exécuté éventuellement, et son stdout ira dans un fichier (au lieu de le voir dans la console)
- Exemple de soumission:

```
$ sbatch ma_tache.sh
```

```
Submitted batch job 123456
```

Script de soumission

```
#!/bin/bash
```

Première ligne obligatoire

```
#SBATCH --account=def-someuser
```

```
#SBATCH --gres=gpu:1      # Request GPU "generic resources"
```

```
#SBATCH --cpus-per-task=6 # Cores proportional to GPUs: 6 on Cedar, 16 on Graham.
```

```
#SBATCH --mem=32000M      # Memory proportional to GPUs: 32000 Cedar, 64000 Graham.
```

```
#SBATCH --time=0-03:00    # DD-HH:MM:SS
```

```
...
```

Commentaires dans le script, qui sont lus par l'ordonnanceur

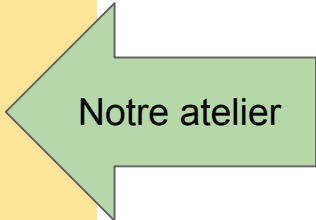
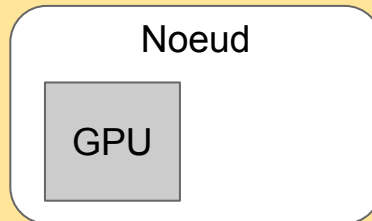
```
python ~/train.py $DATA
```

Lancement de votre processus (e.g. entraînement de modèle)

Parallélisme et GPU

1 noeud, 1 GPU

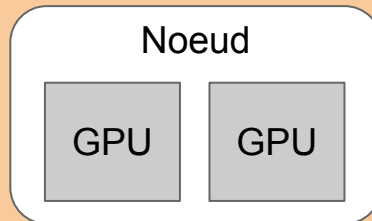
Supporté nativement par les frameworks (v.s. CPU)



Notre atelier

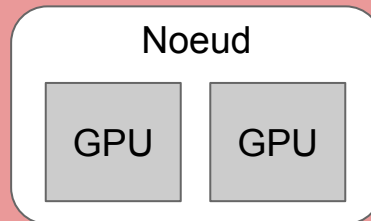
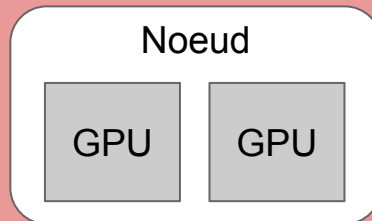
1 noeud, plusieurs GPU

Demande des modifications mineures au code



Plusieurs noeuds, plusieurs GPU

Demande des modifications majeures au code et au script sbatch (selon framework utilisé)



Gestion des données

Gestion des données

Situation: ImageNet sur stockage partagé

- Dataset avec 1 million de petites images
- Stockage partagé centralisé (grappe)
- **Problème:** Accès disque/réseau inefficent
- **Solution:** Encapsuler et transférer au lieu de calcul
- Exemple: `tar` (archiver sans compresser)

Gestion des données

Situation: Énorme base de données, complexe

- Arborescence de données (tenseurs ou autre)
- Utiliser pickle?
- **Problème:** Pas possible de tout charger en RAM
- **Solution:** Utiliser HDF5
- Hierarchical Data Format



Gestion des données

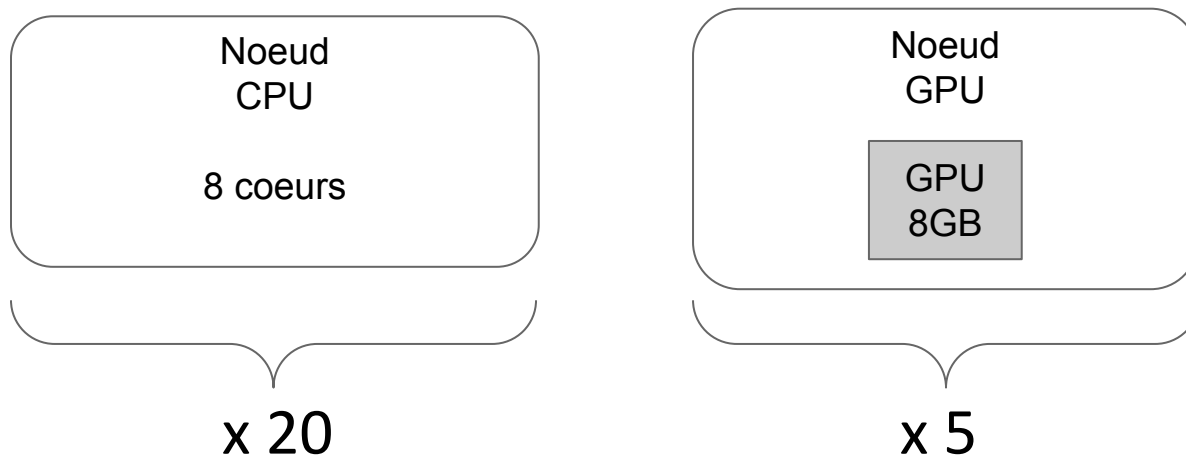
	Encapsulation	Accès partiel
Dossier de fichiers (e.g. jpg)		
tar	✓	
HDF5	✓	✓

Atelier

Atelier

- Objectifs:
 - **Lancer** un entraînement sur une grappe de calcul
 - **Monitorer** une tâche
 - Faire une recherche d'hyperparamètres simple
- Le code d'entraînement est fourni
- Nous utiliserons une mini-grappe temporaire pour fins d'enseignement
- Les instructions complètes de l'atelier sont ici:
github.com/lemairecarl/atelier-dl-cc

La grappe “ift780”



- Il y aura moins de GPUs que d'utilisateurs, vous devrez donc faire la file pour les GPUs (comme IRL)
- Pour lister vos tâches: commande "sq"
- Pour annuler une tâche: commande "scancel <job_id>"

Connexion à la grappe

- [Allez réclamer un username en cliquant ici](#)

Un compte par personne, inscrivez votre nom

- Se connecter à la grappe:

```
ssh username@ift780.calculquebec.cloud
```

Transfert des données et du code

- Base de données TinyImageNet. Vous devrez télécharger le fichier (lien fourni) et l'envoyer sur la grappe.
- Le code est ici: github.com/lemairecarl/atelier-dl-cc
Vous devrez cloner le dépôt sur la grappe.
- Entraînement d'un CNN pour la classification

Environnement Python

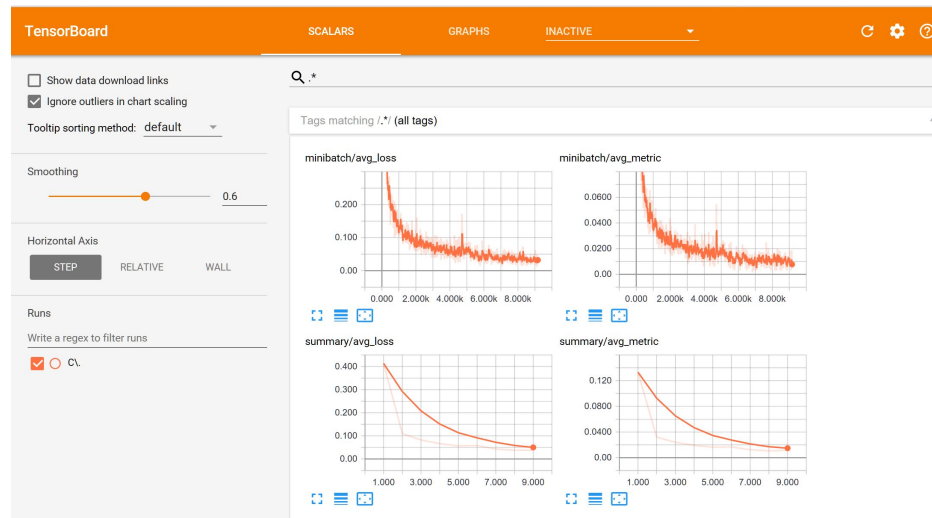
- Vous devrez créer un environnement virtuel
- Installer les paquets Python

Mode interactif v.s. “batch”

- Vous lancerez d’abord l’entraînement en mode interactif
- Ensuite, vous créerez un script qui vous permettra de soumettre une tâche à l’ordonnanceur (mode “batch”)

Suivre la tâche

- Vous afficherez l'output d'une tâche, contenu dans un fichier
- Vous suivrez l'entraînement avec Tensorboard



Recherche d'hyperparamètres

- Vous ferez une recherche d'hyperparamètres très simple
- Plusieurs tâches **peuvent** rouler en parallèle
- Vous comparerez les résultats de la recherche dans TensorBoard

Pour aller plus loin

Il vous est recommandé de suivre ce tutoriel sur PyTorch:

https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Ces tutoriels sur TensorFlow pourraient aussi vous intéresser:

<https://www.tensorflow.org/tutorials/quickstart/beginner>

<https://www.tensorflow.org/tutorials/keras/classification>

<https://www.tensorflow.org/guide/gpu>