# Machine learning : comment AWS aborde l'entraînement distribué

Distribuer le modèle ou les données ? C'est l'une des questions inhérentes à l'entraînement de modèles d'apprentissage automatique. En particulier de *deep learning*. AWS y a consacré plusieurs annonces dans le cadre de sa conférence re:Invent, en cours jusqu'au 18 décembre.

L'*hyperscaler* américain a dans sa « boîte à outils » SageMaker deux bibliothèques logicielles qui permettent d'implémenter l'une et l'autre approche. Elles couvrent pour le moment les *frameworks* PyTorch et TensorFlow.

La méthode la plus commune pour l'entraînement distribué consiste à faire travailler un modèle sur des sous-ensembles (*mini-batchs*) d'un jeu de données. Et d'estimer à chaque fois un gradient d'erreur qui permet de mettre à jour le coefficient.

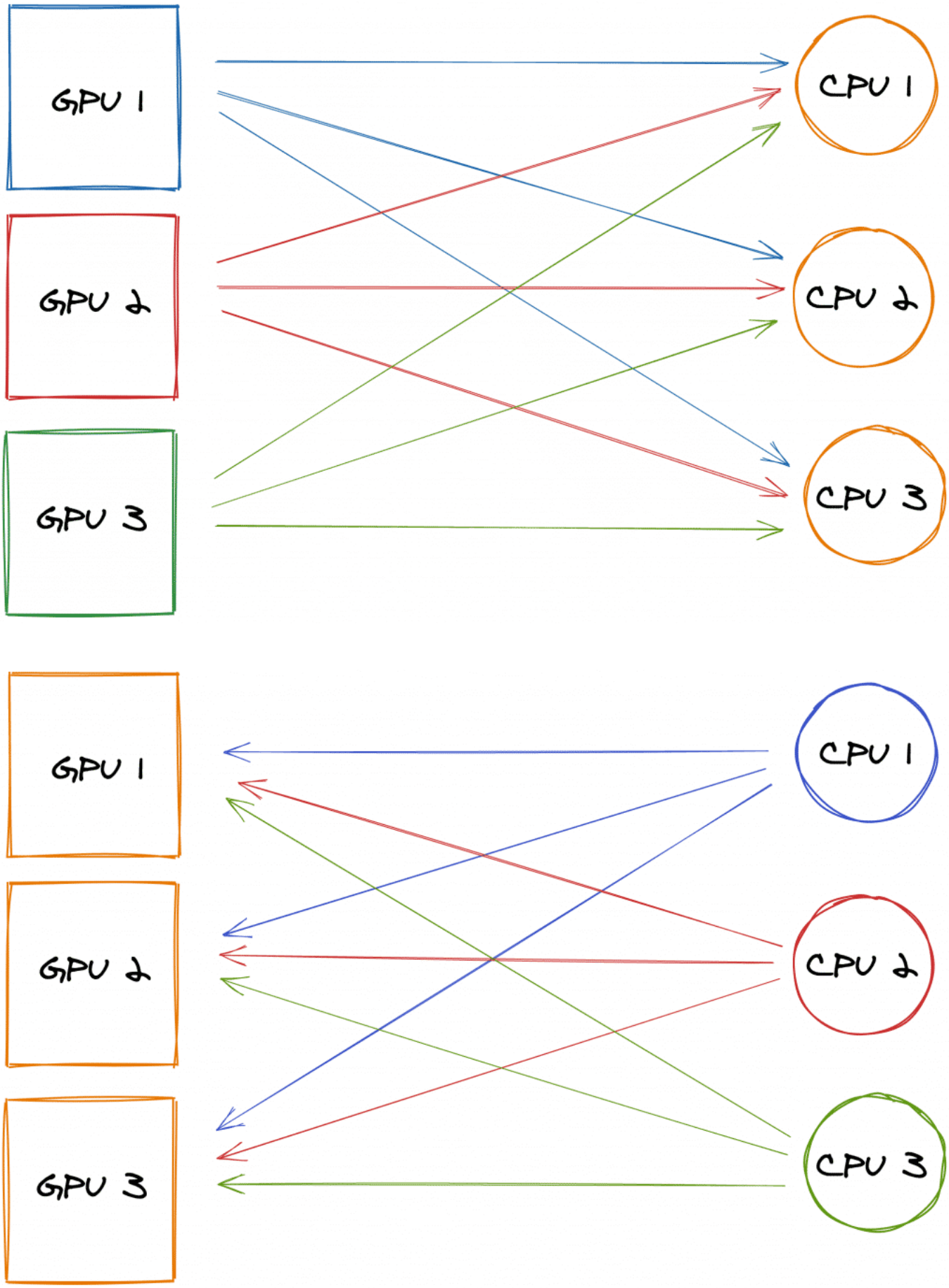
Sur le papier, ce modèle itératif assure une croissance linéaire du temps d'entraînement, fonction de la taille du jeu de données. Surtout, il se prête à la parallélisation, le modèle traitant individuellement chaque élément des *mini-batchs*.

## Partitionner les données…

Généralement, on commence par distribuer les sous-ensembles de données. Cela a pour effet d'accélérer leur traitement. Mais fait aussi naître une difficulté : il faut centraliser les gradients de toutes les tâches qui composent chaque *mini-batch*. Ce processus, nommé *allreduce*, prend d'autant plus d'empreinte que le cluster d'entraînement grandit.

Le projet Horovod, aujourd'hui intégré aux principales bibliothèques de *machine learning*, implémente un algorithme asynchrone appelé ring-allreduce. L'image de l'anneau (*ring*) reflète son fonctionnement : cyclique et directionnel. Chaque nœud reçoit le gradient de son « prédécesseur », l'applique en même temps que celui qu'il a lui-même calculé, puis transmet le tout à son voisin. Un tel système a l'avantage de réduire les communications entre nœuds. Pour n GPU, chacun traite en l'occurrence 2 x (n-1) messages.

AWS s'appuie sur ce modèle, mais va plus loin, en supprimant toute communication directe entre les GPU. Chacun stocke son gradient en mémoire et, passé un certain palier de capacité, le divise en autant de parties qu'il existe de serveurs de paramètres. Ces serveurs sont pris en charge par les CPU des instances d'entraînement, qui centralisent les gradients et les redistribuent à tous les GPU. Tous participent de manière égale (ils traitent le même volume de données).


GPU 1
GPU 2
GPU 3
CPU 1
CPU 2
CPU 3
GPU 1
GPU 2
GPU 3
CPU 1
CPU 2
CPU 3

# … ou les modèles

Il arrive qu'un modèle ne puisse se satisfaire d'un seul CPU ou GPU (GPT-3, le dernier-né d'OpenAI, entre dans cette catégorie avec ses 175 milliards de paramètres). Dans ce cas, on peut tenter d'en modifier les hyperparamètres, de réduire la taille des *mini-batchs* ou encore de compresser le gradient. Mais parfois, il faut distribuer le modèle lui-même… et envoyer à chacune de ses parties tous les *mini-batchs*.

Cette approche, sérialisée par essence, entraîne souvent une sous-utilisation des GPU. Les principaux *frameworks* de *machine learning* y ont apporté une solution partielle à travers l'exécution planifiée. Pour aller un peu plus vers le parallélisme, AWS reprend cette brique et y associe une architecture de *micro-batchs* entrelacés. Il y ajoute une option de partitionnement automatique des modèles, avec deux critères au choix : privilégier la vitesse ou les économies de mémoire.

| **re:Invent 2020 : ode à SageMaker** |
| --- |
| AWS a multiplié, ces derniers jours, les annonces relatives à SageMaker. Parmi elles :<br>• Extension du débogueur, avec la possibilité de suivre l'usage des ressources système (fonctionne avec PyTorch et TensorFlow)<br>• Clarify, une marque qui regroupe des outils dédiés à la détection des biais et à la compréhension des résultats que fournissent les modèles<br>• Edge Manager, une offre de gestion du *machine learning* en périphérie<br>• Feature Store, un *datastore* destiné à centraliser les paramètres utilisés pour la préparation des données<br>• Jumpstart, qui donne accès à un catalogue de modèles et de *workflows* prêts à l'emploi<br>• Pipelines, pour la mise en place du CI/CD<br>• Data Wrangler, une interface visuelle pour la préparation de données |

*Illustration principale © psdesign1 – Fotolia*