LIVRE BLANC TECHNIQUE

L'INTELLIGENCE ARTIFICIELLE DANS LA PRATIQUE

LE DEEP LEARNING (APPRENTISSAGE PROFOND) ALIMENTÉ PAR FLASHBLADE

PURESTORAGE®

TABLE DES MATIÈRES

INTRODUCTION

Les progrès réalisés dans les réseaux neuronaux profonds ont généré une nouvelle vague d'algorithmes et d'outils d'intelligence artificielle (IA) permettant d'exploiter les données. Grâce à ces algorithmes améliorés, des jeux de données plus vastes et des environnements informatiques comme TensorFlow, les spécialistes des données peuvent s'attaquer à de nouvelles applications, comme les véhicules autonomes ou le traitement du langage naturel.

L'entraînement des réseaux neuronaux profonds nécessite deux choses : des données de qualité et une importante capacité de calcul. Les processeurs graphiques, ou GPU, sont des processeurs massivement parallèles capables de traiter simultanément d'importants volumes de données. Lorsque plusieurs GPU sont regroupés en cluster, un pipeline haut débit est nécessaire pour extraire du stockage les données d'entrée qui alimenteront les moteurs de calcul. Mais l'apprentissage profond ne se résume pas à la construction et à l'entraînement de modèles. Il s'accompagne de tout un pipeline de données qui doit permettre les mises à l'échelle, les itérations et les expérimentations sans lesquelles l'équipe de spécialistes des données ne peut mener à bien sa mission.

> Le présent document expose la justification technique et les avantages d'un système d'entraînement de bout en bout, et les raisons de l'importance de la plate-forme Pure Storage® FlashBlade™. Il montre également les tests de performances réalisés sur un système composé à la fois de serveurs NVIDIA® DGX-1™ équipés de plusieurs GPU et spécialement conçus pour les applications d'apprentissage profond (deep learning), et de Pure FlashBlade, une plate-forme de données dynamique, évolutive et performante capable d'alimenter l'ensemble du pipeline de données d'IA.

CYCLE DE VIE DES DONNÉES

Ce sont les données qui alimentent les algorithmes modernes d'intelligence artificielle et d'apprentissage profond. Avant de démarrer l'entraînement d'un modèle d'IA, il faut s'atteler à la collecte des données étiquetées sans laquelle le modèle manquera de précision. Un système d'IA déployé à taille réelle doit collecter, nettoyer, transformer, étiqueter et stocker en permanence de gros volumes de données. L'ajout de points de données supplémentaires, s'ils sont de qualité, se traduit directement par un modèle plus précis.

Les échantillons de données subissent différentes étapes de traitement :

- **INGESTION**, par le système d'entraînement, de données provenant d'une source extérieure. Un point de données est le plus souvent un fichier ou un objet. Un moteur d'inférence peut également s'exécuter sur ces données. Après l'étape d'ingestion, les données brutes sont stockées et souvent sauvegardées, également sous leur forme brute. Les étiquettes associées (réalité de terrain) peuvent être acheminées avec les données ou suivre un flux d'ingestion distinct.
- **NETTOYAGE ET TRANSFORMATION** des données, enregistrement dans un format pratique pour l'entraînement et liaison de l'échantillon de données et de l'étiquette correspondante. Cette seconde copie des données n'est pas sauvegardée car elle peut être recalculée, si nécessaire.
- **EXPLORATION** des paramètres et des modèles, test rapide sur un jeu de données réduit et itération pour déterminer les modèles les plus intéressants à transmettre au cluster de production.
- **ENTRAÎNEMENT** par phases, avec sélection aléatoire de lots de données d'entrée (échantillons anciens et nouveaux), et transfert pour calcul par les GPU des serveurs de production avant mise à jour les paramètres du modèle.
- **ÉVALUATION** sur une partie des données qui est retenue et non utilisée pour l'entraînement, afin d'évaluer la précision du modèle.

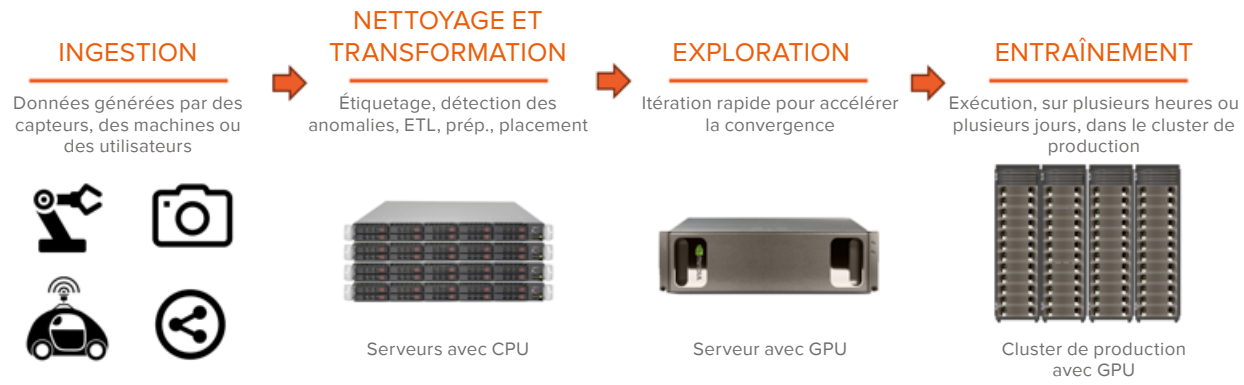
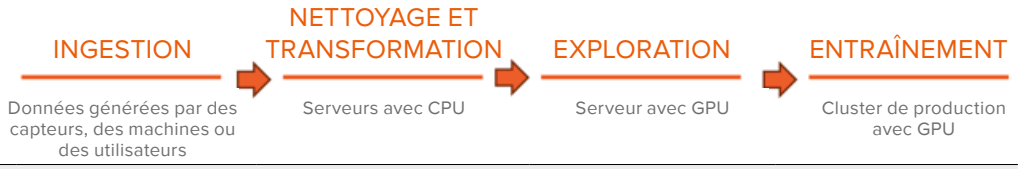


FIGURE 1 : Étapes de traitement d'un échantillon de données d'IA

Ce cycle de vie ne concerne pas uniquement les réseaux neuronaux ou l'apprentissage profond, mais tout type d'apprentissage machine en parallèle. Ainsi, les environnements standard d'apprentissage machine comme Spark MLlib utilisent des processeurs classiques (CPU) à la place de processeurs graphiques (GPU), mais les processus d'ingestion des données et d'entraînement sont les mêmes.

> La coordination de la totalité du cycle de vie se fait sur une seule plate-forme de stockage partagé, sans que des copies supplémentaires des données soient nécessaires pour les étapes d'ingestion, de prétraitement et d'entraînement. Il est rare que les données ingérées soient utilisées pour une seule opération, et le stockage partagé permet d'entraîner plusieurs modèles différents ou d'appliquer aux données des outils d'analyse classique.



Modèle d'accès	Séquentiel	Séquentiel ou aléatoire	Aléatoire	Aléatoire
Type d'accès	Écriture	Lecture ou écriture	Lecture	Lecture
Taille de fichier	Petit volume de métadonnées Petit à gros volume de métadonnées	Petit à gros	Petit à gros	Petit à gros
Concurrence	Variable selon le nombre de sources	Élevée	Faible	Élevée

FIGURE 2 : Besoins en stockage du pipeline d'IA

Comme nous l'avons vu précédemment, chaque étape du pipeline d'IA a des besoins différents par rapport à la plate-forme de données. Les systèmes de stockage à évolutivité horizontale (scale-out) doivent garantir des performances sans compromis, quels que soient le type ou le schéma d'accès (petits fichiers incluant de nombreuses métadonnées ou gros fichiers, schémas d'accès aléatoires ou séquentiels, concurrence faible ou élevée). Or, cette condition ne peut être remplie avec les systèmes de stockage traditionnels. En revanche, parce qu'elle a été conçue de A à Z pour les charges de travail modernes et non structurées, la plate-forme de données FlashBlade est parfaitement adaptée à l'IA.

Dans l'idéal, à la première étape, les données sont ingérées et stockées sur la plate-forme qui sera utilisée pour les étapes suivantes, pour ne pas multiplier les copies des données. Les deux étapes suivantes peuvent être réalisées sur un serveur de calcul standard, équipé ou non d'un processeur graphique. Ensuite, pour la quatrième et dernière étape, les tâches complètes d'entraînement en environnement de production sont exécutées sur des serveurs puissants accélérés par GPU, comme le DGX-1. Il n'est pas rare qu'un pipeline de production et un pipeline expérimental s'exécutent en parallèle sur un même jeu de données. De plus, les GPU du système DGX-1 peuvent être utilisés chacun pour un modèle différent, ou entraîner ensemble un modèle plus large. Plusieurs systèmes DGX-1 peuvent même être utilisés pour l'entraînement distribué.

Si la couche de stockage partagé est lente, les données doivent être copiées sur un stockage local pour chacune des étapes. On perd alors beaucoup de temps à déployer les données sur différents serveurs. La plate-forme de données idéale pour le pipeline d'entraînement IA est celle dont les performances sont les mêmes que si les données étaient stockées localement sur un nœud du serveur, et qui est suffisamment simple et performante pour que toutes les étapes du pipeline puissent s'exécuter de façon concomitante.

FLUX DE TRAVAIL DU SCIENTIFIQUE DES DONNÉES

Les scientifiques des données (data scientists) ont pour mission de renforcer par différents moyens - multiplication ou amélioration des données, entraînement plus rationnel, modèles approfondis - la validité du modèle entraîné. Le plus souvent, ils sont regroupés en équipes qui partagent les mêmes jeux de données et travaillent en parallèle pour produire de nouveaux modèles d'entraînement plus performants.

Le flux de travaux quotidien d'un scientifique des données se compose des tâches suivantes :

- Classement, nettoyage, filtrage, traitement et transformation des données d'entraînement sous une forme utilisable par le modèle à entraîner.
- Expérimentation, test et débogage d'un modèle sur un sous-ensemble réduit des données d'entraînement.
- Entraînement du modèle avec l'ensemble complet de données d'entraînement, sur des durées plus longues.

> Ce workflow se fait par itérations entre les étapes de développement, d'expérimentation et de débogage. Le principal outil de développement est un environnement d'apprentissage profond, par exemple TensorFlow, Caffe2, CNTK, etc. Ces environnements incluent des utilitaires de traitement des données et de création de modèles optimisés pour s'exécuter sur des GPU distribués.

Souvent, une équipe de spécialistes des données travaille simultanément sur ces étapes avec des jeux de données partagés. L'existence de plusieurs charges de travail concurrentes de traitement des données, d'expérimentation et d'entraînement grandeur nature nécessite des schémas d'accès multiple à la couche de stockage. Autrement dit, le stockage ne doit pas seulement permettre la lecture de gros fichiers, mais également pouvoir prendre en charge un mélange de petits et gros fichiers en lecture et en écriture.

Enfin, si plusieurs spécialistes des données explorent les jeux de données et les modèles, il est essentiel d'enregistrer les données dans leur format d'origine pour qu'ils puissent chacun transformer, nettoyer et utiliser les données de la manière qui lui convient. En définitive, ce sont ces expérimentations qui donnent naissance aux modèles les plus performants.

FlashBlade est une plate-forme de stockage partagé naturellement adaptée aux jeux de données. Elle offre des fonctions de redondance double parité pour la protection des données ainsi que les performances nécessaires pour servir de point d'accès commun à plusieurs développeurs et à de multiples expériences. FlashBlade évite d'avoir à copier avec soin des sous-ensembles de

données pour y travailler localement : c'est du temps gagné pour les ingénieurs, mais aussi en utilisation du système DGX-1. De plus, ces copies deviennent de plus en plus coûteuses, les jeux de données brutes et les transformations souhaitées étant constamment mis à jour et modifiés.

DES JEUX DE DONNÉES ÉVOLUTIFS

L'une des raisons fondamentales qui expliquent le succès de l'apprentissage profond tient à l'amélioration constante des modèles liée à l'augmentation de la taille des jeux de données. A contrario, les algorithmes classiques d'apprentissage machine, par exemple les algorithmes de régression logistique, cessent de gagner en précision avec l'augmentation des tailles de jeux de données.

Un éminent chercheur, spécialiste de l'IA, évoquait récemment cette problématique :

« Depuis 2016, en règle générale, un algorithme supervisé d'apprentissage profond atteint généralement une performance acceptable avec environ 5000 exemples étiquetés par catégorie, et une performance supérieure ou égale à celle d'un humain lorsqu'il s'entraîne sur un jeu de données contenant au moins dix millions d'exemples étiquetés. »
Ian Goodfellow 2016.

> Des recherches récentes menées par Google montrent l'intérêt d'augmenter la taille des jeux de données, avec un accroissement logarithmique des performances des tâches de reconnaissance visuelle lorsque les jeux de données atteignent 300 millions d'images. Ces recherches suggèrent même que les modèles à grande capacité nécessitent des jeux de données proportionnellement plus vastes.

La séparation des tâches de calcul (DGX-1) et de stockage (FlashBlade) permet également de dimensionner chaque couche indépendamment, ce qui évite certaines difficultés liées à la gestion commune des deux couches. Un système de stockage scale-out doit pouvoir grandir facilement si la taille des jeux de données augmente ou si l'on envisage d'utiliser de nouveaux jeux de données. De même, si davantage d'entraînements concurrents sont nécessaires, il est possible d'ajouter des GPU ou des serveurs DGX-1 supplémentaires sans se soucier du stockage interne.

POURQUOI CHOISIR FLASHBLADE

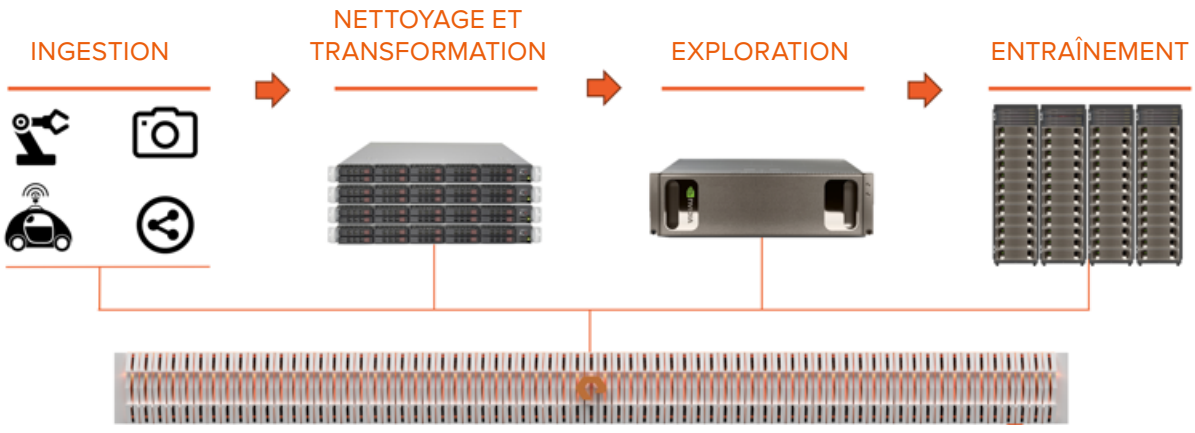


FIGURE 3 : FlashBlade comme plate-forme de données centralisée

Dans une architecture d'apprentissage profond, une plate-forme de données centralisée augmente l'efficacité des scientifiques des données et simplifie le dimensionnement et l'exploitation du système. FlashBlade, en particulier, facilite la création, l'exploitation et l'extension du système d'IA, et ce pour différentes raisons.

- **PERFORMANCE :** avec plus de 15 Go/s de bande passante en lecture aléatoire par châssis pour un total de 75 Go/s, la plate-forme FlashBlade peut prendre en charge des exigences concurrentes sur la totalité du processus d'IA.
- **PRISE EN CHARGE DES FICHIERS DE PETITE TAILLE :** parce qu'elle peut lire de façon aléatoire des petits fichiers (de 50 Ko) à 10 Go/s sur un même châssis FlashBlade (50 Go/s avec 75 lames), FlashBlade n'exige aucun effort supplémentaire pour agréger des points de données individuels afin de créer des fichiers plus volumineux et plus adaptés au stockage.
- **ÉVOLUTIVITÉ :** il est possible de démarrer avec un système modeste puis d'ajouter une lame pour accroître les capacités et les performances, si le jeu de données se développe ou si le débit doit être augmenté.
- **PRISE EN CHARGE DES OBJETS NATIFS (S3) :** les données d'entrée peuvent être stockées sous forme de fichiers ou d'objets.
- **SIMPLICITÉ D'ADMINISTRATION :** inutile d'optimiser les performances en fonction de la nature des fichiers, ou de "re-paramétrer" le stockage lors de l'augmentation du système de fichier.
- **MISES À NIVEAU TRANSPARENTES INTÉGRALES :** les mises à niveau logicielles et l'extension matérielle peuvent se faire à tout moment, y compris pendant l'entraînement des modèles en production.
- **SIMPLICITÉ DE GESTION :** avec Pure1®, notre plateforme de gestion et de support en mode cloud, les utilisateurs peuvent suivre leur stockage depuis l'appareil de leur choix et bénéficient d'un support prédictif qui permet d'identifier et de résoudre les problèmes avant qu'ils n'aient une incidence. Ils peuvent ainsi se concentrer sur la compréhension des données plutôt que sur l'administration du stockage.
- **UN SYSTÈME À L'ÉPREUVE DU TEMPS :** spécialement conçu pour la technologie flash, il permet d'exploiter facilement les nouvelles générations de mémoire NAND, synonymes de densité, de faible coût et de rapidité.

Les performances du stockage avec les fichiers de petite taille sont essentielles, de nombreux types d'entrées (texte, son ou image) étant stockés sous leur forme originelle de petits fichiers. Si le stockage ne gère pas correctement ces petits fichiers, il faut ajouter une étape supplémentaire pour le prétraitement et le regroupement des échantillons en fichiers plus importants.

Lorsque le système de stockage s'appuie sur des disques rotatifs, avec une couche de mise en cache reposant sur des disques SSD, il n'atteint pas le niveau de performances requis. Comme l'entraînement avec des inputs aléatoires donne des modèles plus précis, l'ensemble du jeu de données doit être accessible et bénéficier de performances optimales. Les caches SSD n'offrent des performances élevées que pour une petite partie des données, et ne parviennent pas à compenser la latence des disques rotatifs.

Enfin, grâce à ses performances et à son niveau de concurrence, FlashBlade permet aux spécialistes des données de passer rapidement d'une étape à l'autre de leur travail, sans perdre de temps à copier les données. FlashBlade permet également l'exécution simultanée d'expériences multiples sur les mêmes données.

ARCHITECTURE DU SYSTÈME

Nous décrirons dans cette section les composants matériels et logiciels nécessaires dans un cluster d'apprentissage profond.

Composants d'infrastructure

L'infrastructure physique d'un cluster d'apprentissage profond basé sur des serveurs DGX-1 et une plate-forme FlashBlade inclut les équipements suivants :

Capacité de calcul principale : Serveur(s) DGX-1 équipé(s) de huit GPU P100 (ou, bientôt, de GPU Volta V100) et interconnexion par NVLink à un moteur de calcul permettant d'entraîner les paramètres pour les réseaux neuronaux profonds. Chaque système dispose d'une connexion Ethernet et InfiniBand externe. Les GPU peuvent être regroupés en cas d'entraînement unique de grande ampleur, ou utilisés séparément pour l'entraînement de modèles multiples.

Stockage : FlashBlade est un système évolutif 100 % flash où sont stockés les fichiers ou les objets, avec une capacité utilisable (pour les données non compressibles) comprise entre 60 To et 2,5 Po. Les données sont accessibles à travers les protocoles hautes performances NFS ou S3.

Gestion du réseau : la redondance est assurée par des switches Ethernet ToR (top-of-rack) redondants, connectés aux systèmes de stockage et de calcul à travers des ports de 10 et 40 Gbps sur des canaux MLAG Port Channel. Les systèmes DGX-1 sont généralement reliés entre eux par un réseau InfiniBand. Toutefois, certains clients préfèrent utiliser la même réseau Ethernet pour le trafic de calcul et le trafic de stockage.

Calcul supplémentaire : serveurs white-box, éventuellement avec GPU, pour l'ingestion et le prétraitement des données et le débogage des modèles.

Voir plus loin la section consacrée au dimensionnement pour savoir comment déterminer le rapport entre le nombre de plates-formes FlashBlade et de serveurs DGX-1.

Pipeline logiciel

Le logiciel d'entraînement doit acheminer les données du système de stockage vers les GPU, et veiller à ce que les GPU aient toujours à leur disposition le lot suivant de données d'entraînement.

Pour une session complète d'entraînement d'un modèle, les données passent par les étapes suivantes :

- **DÉCODAGE ET AUGMENTATION :** chargement des fichiers d'entrée à partir du support de stockage et conversion sous une forme adaptée à l'entraînement. L'étape de décodage peut être prétraitée ou exécutée sur le processeur hôte. L'augmentation utilise le processeur hôte pour introduire des modifications dynamiques dans les fichiers d'entrée et éviter les risques de surapprentissage.
- **FILES D'ATTENTE D'E/S :** plusieurs processus lisent des lots aléatoires d'enregistrements d'entrée provenant du stockage, et remplissent une file d'attente interne. Ces processus, qui s'exécutent sur les CPU hôtes du DGX-1, sont chargés de récupérer les lots à l'avance et de s'assurer que les enregistrements d'entraînement sont disponibles dans la DRAM.
- **ENTRAÎNEMENT :** une seconde série de processus extrait les données des files d'attente internes et alimente les GPU chargés des calculs (par exemple, propagation avant et rétropropagation) nécessaires pour entraîner et mettre à jour les paramètres des modèles.

Pour chaque lot d'entraînement, les données doivent être extraites du stockage permanent, puis décodées et augmentées.
Si les GPU sont inactifs (s'ils attendent des E/S ou que le CPU hôte récupère et décode les entrées), leur puissance de calcul n'est

pas efficacement exploitée. Le flux d'entraînement illustré ci-après garantit que les systèmes d'entraînement disposent toujours du jeu d'entrée suivant dès que le lot précédent est terminé. Tant que la portion E/S et CPU hôte est plus rapide que les calculs d'apprentissage, les GPU fonctionnent à plein régime.

Pipeline d'entraînement classique

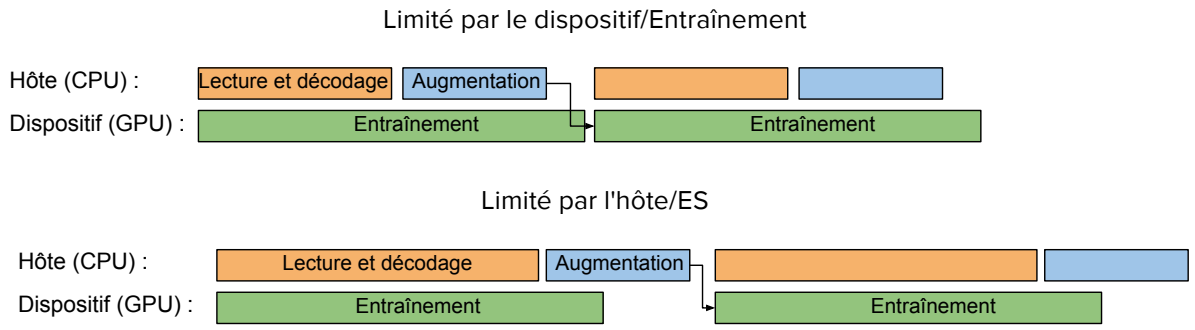


FIGURE 4 : Flux des données d'entraînement dans l'architecture logicielle

Le schéma ci-après illustre le mouvement des données d'entraînement à travers l'architecture logicielle pour assurer le traitement en pipeline décrit. Les CPU hôtes du DGX-1 orchestrent deux séries de processus asynchrones. Les processus de récupération choisissent aléatoirement des lots de fichiers d'entrée et sont chargés de placer les données dans une file d'attente si celle-ci n'est pas encore pleine. Les processus d'entraînement exécutent le modèle d'entraînement et retirent les données d'entrée de la file d'attente à chaque itération. Le point de coordination des deux processus est la file d'attente en mémoire. Les processus de récupération peuvent également prétraiter les données d'entrée et les mettre au format attendu. Tant que le débit des E/S et du CPU hôte est suffisant pour éviter que la file d'attente ne se vide, l'exécution des processus d'entraînement peut se faire à une vitesse optimale.

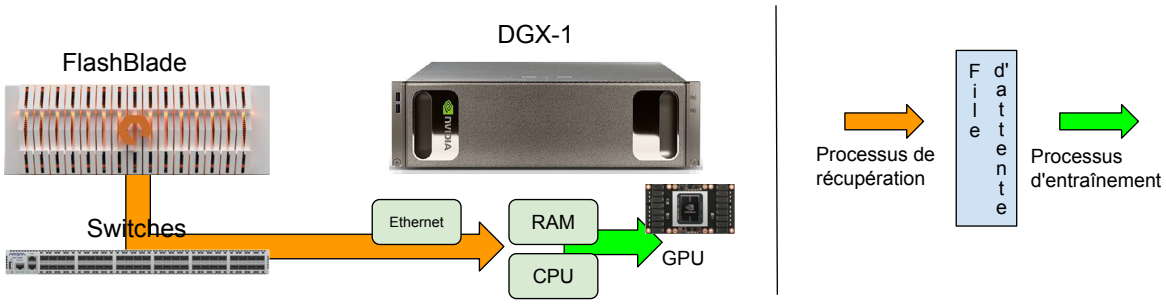


FIGURE 5 : La rapidité de FlashBlade garantit une vitesse d'entraînement optimale

Vous trouverez dans la documentation de TensorFlow des informations plus détaillées sur le traitement en pipeline de la lecture.

Connectivité

La connexion physique de FlashBlade se fait par Ethernet ; les serveurs DGX-1 sont équipés de ports Infiniband (IB) et Ethernet. FlashBlade se connecte aux switches ToR via un maximum de huit ports à 40 Gbps dans un même MLAG, présentant une seule liaison logique. Tous les accès au stockage se font via ces liaisons Ethernet et le protocole NFS v3, selon les options de montage par défaut ; rsize et wsize ont une taille de 512 Ko, et la mise en cache du système de fichiers n'est pas nécessaire.

Les ports IB du système DGX-1 permettent d'interconnecter plusieurs serveurs DGX-1 dans un réseau à faible latence, afin d'augmenter la capacité d'entraînement ; ils peuvent aussi être raccordés au réseau Ethernet pour augmenter la bande passante vers le stockage externe. Les interconnexions NVLink internes permettent de relier entre eux plusieurs GPU et de coordonner les mises à jour des paramètres des modèles après chaque itération d'entraînement. Pour étendre le système DGX-1, il est possible de connecter de nouveaux serveurs DGX-1 via les ports IB et d'organiser l'entraînement sur plusieurs serveurs.

Avec les modèles et les GPU actuels, les deux ports Ethernet à 10 Gbps assurent une bande passante suffisante pour alimenter les GPU en données. Si l'entraînement exige une bande passante supérieure à celle d'une liaison unique de 10 Gbps, il existe deux possibilités pour configurer les ports Ethernet des serveurs DGX-1 :

- associer les deux liaisons et fournir une connexion logique unique de 20 Gbps aux commutateurs et à FlashBlade,
- affecter chaque port à un sous-réseau différent et le relier à deux VIP (adresses IP virtuelles) de données FlashBlade sur le sous-réseau correspondant.

> Dans les deux cas, l'objectif est d'utiliser la totalité de la bande passante Ethernet grâce au multiplexage de la lecture des données d'entrée sur les deux liaisons.

Chaque session d'entraînement est reliée (et crée une connexion TCP) à un point de montage unique. Que les ports soient associés ou séparés, la connexion TCP utilise une seule des deux liaisons. Nous utilisons sur FlashBlade plusieurs VIP de données pour monter un même système de fichiers sur plusieurs points de montage ; comme chaque montage est connecté à une adresse IP différente, le kernel crée pour chacun une connexion TCP différente.

RÉSULTATS DES TESTS DE PERFORMANCES DE L'ENTRAÎNEMENT

La section suivante présente les résultats des tests de performance sur les opérations d'entraînement avec un système complet FlashBlade + DGX-1. Les performances sont mesurées avec les modèles ImageNet courants, le but étant de s'assurer que **les données lues directement depuis FlashBlade occupent entièrement les GPU**. Avec l'amélioration des GPU (notamment avec l'architecture V100) et la possibilité d'ajouter des serveurs DGX-1 supplémentaires, les résultats ci-après nous aident aussi à évaluer le nombre de GPU qu'une FlashBlade peut prendre en charge.

Configuration des tests

La série de tests d'entraînement a été réalisée avec le matériel DGX-1 et FlashBlade décrit à la section Architecture du système. Le système DGX-1 est équipé de huit GPU P100, de deux CPU hôtes (pour un total de 80 cœurs) et de 512 Go de mémoire ; les données d'entraînement sont stockées sur une plate-forme FlashBlade de 15 lames. Tous les tests ont été exécutés avec container fourni par NVIDIA : nvcr.io/nvidia/tensorflow v17.07.

Le jeu de données utilisé pour l'entraînement est Imagenet 2012, qui occupe une place prépondérante dans les recherches sur l'apprentissage profond. Les données sont des images jpeg étiquetées, rassemblées dans des fichiers plus importants (d'environ 135 Mo chacun). L'option de mise en cache du système de fichiers (fsc) est désactivée sur le montage NFS. Les meilleures pratiques couramment appliquées à l'optimisation des performances de TensorFlow sont respectées, et chaque modèle est entraîné jusqu'à ce que le nombre d'images traitées par seconde se stabilise.

Si ImageNet joue un rôle important pour la communauté de l'apprentissage profond, la taille et le format des fichiers ne sont pas ceux des données utilisées dans la réalité. Les données ImageNet sont prétraitées et rassemblées dans des fichiers volumineux, alors que l'on préfère utiliser les formats d'origine (> 1 Mo) garantissant une fidélité parfaite afin d'accélérer le prototypage de nouvelles techniques susceptibles d'améliorer l'exploitation des données.

Pour les tests de performances, nous comparons trois sources d'entrée différentes des données d'entraînement :

- Des données synthétiques sont créées artificiellement dans la mémoire système. Il n'y a donc aucune opération d'E/S au niveau du stockage ou de décodage des images au format jpeg. Le but de cette configuration est d'identifier les limites du traitement par les GPU pour comprendre si ceux-ci peuvent constituer un goulet d'étranglement.
- Quatre disques SSD locaux, situés à l'intérieur du boîtier DGX-1, contiennent le jeu de données d'entraînement. Dans ce scénario, le jeu de données est mis en cache dans les SSD locaux, sans quoi il devrait être placé sur le DGX-1 depuis un autre référentiel de stockage externe. Dans cette méthode, la capacité est limitée à la capacité locale (8 To actuellement).
- FlashBlade avec 15 lames est utilisée comme source des données d'entraînement. Les données sont lues depuis FlashBlade via des connexions TCP multiples. La capacité de FlashBlade peut atteindre 2,5 Po, avec des performances identiques sur l'ensemble du jeu de données.

> L'objectif de ces tests de performances est de vérifier que la latence de la liaison avec le stockage interne ne crée pas de goulet d'étranglement.

Pour utiliser les deux liaisons de 10 Gbps pour connecter le système DGX-1 à FlashBlade, il a fallu accéder aux données avec des connexions TCP multiples. Le système de fichiers FlashBlade a été monté sur deux VIP différents et les accès en lecture aux fichiers ont été multiplexés sur les deux points de montage pendant la phase d'entraînement.

Résultats

Une ligne de référence a été établie en comparant le débit en lecture de deux options de stockage : SSD internes et FlashBlade. Débit en lecture de données aléatoires mesuré à partir du système DGX-1 :

- 4 SSD locaux RAID-0 : 2,09 Go/s
- FlashBlade : 2,14 Go/s pour deux liaisons à 10 Gbps

Avec les deux interfaces 10 Gbps, la bande passante en lecture était équivalente pour le stockage local et FlashBlade. Cette performance en lecture de 2 Go/s représente la limite supérieure de vitesse des E/S pour chacune des deux configurations, et laisse des performances disponibles pour les autres nœuds de calcul (ingestion des données, expérimentation et débogage des modèles).

Résultats des tests de performances sous TensorFlow

Les tests de performances sous TensorFlow sont exécutés sur plusieurs itérations afin d'obtenir une mesure stable du nombre d'images traitées chaque seconde pendant l'entraînement. Chaque test est exécuté avec un nombre variable de GPU, en mode « data parallel », avec optimisation par SGD (descente de gradient stochastique).

Un taux élevé d'images par seconde signale des besoins plus importants en vitesse de traitement et en débit. On mesure les images par seconde plutôt que le débit car chaque modèle doit avoir des entrées de taille fixe, quelle que soit la taille de l'image d'origine. Chaque pipeline inclut une étape de redimensionnement avant l'envoi des données aux GPU ; les fichiers d'entrée volumineux représentent une charge plus importante pour le stockage mais pas pour les GPU.

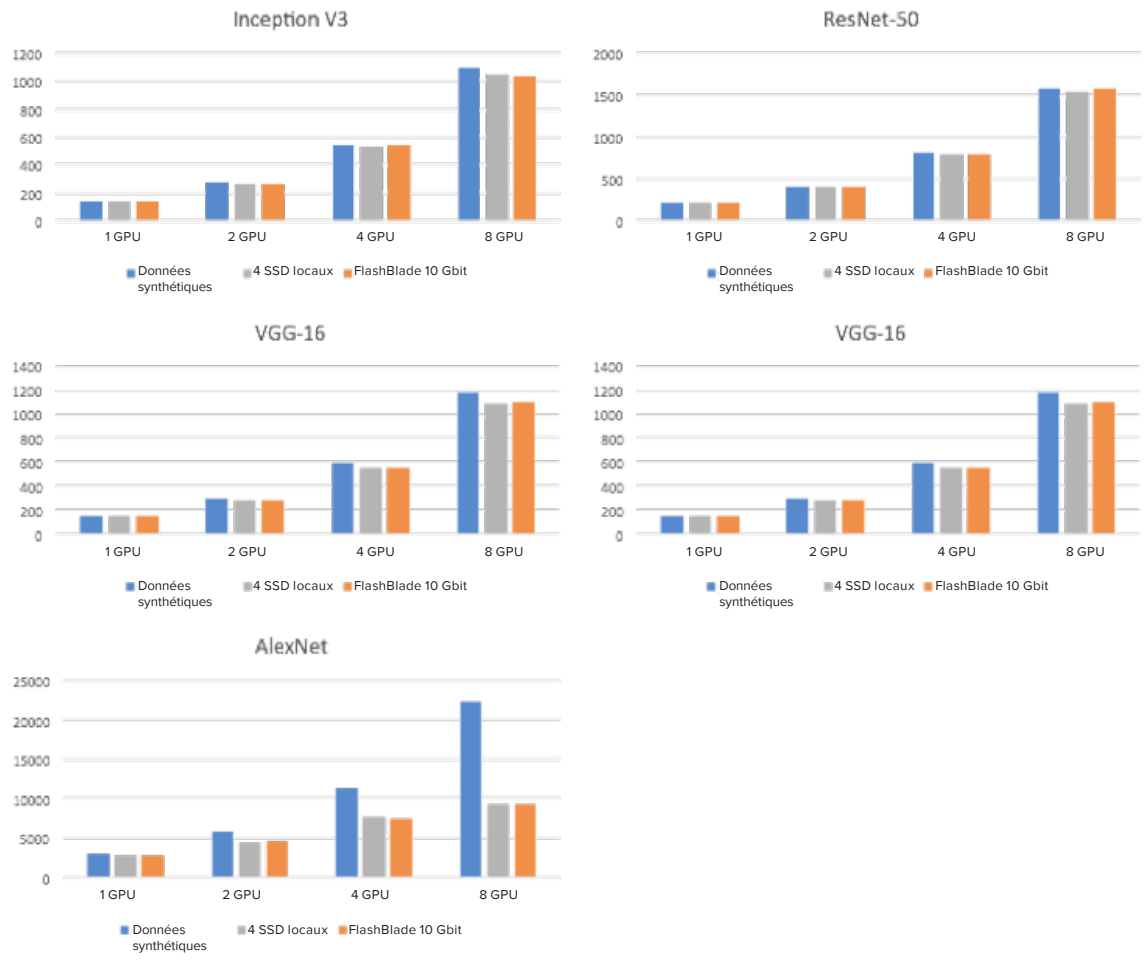


FIGURE 6 : Résultats des tests de performances sous TensorFlow, par modèle. AlexNet est dépendant des CPU en raison de la transformation des images. Tous les autres modèles sont dépendants des GPU.

> Tous les modèles plus complexes (Inception-v3, ResNet-50, Resnet-152 et VGG-16) sont ralentis au niveau des GPU. En témoigne le niveau des performances d'entraînement équivalent avec les SSD locaux et FlashBlade, ce qui signifie que la file d'attente in-memory est suffisamment pleine pour assurer les débits exigés par les GPU. La charge de travail synthétique est légèrement plus rapide, ce qui semble indiquer que les transformations d'images sur le CPU ne représentent qu'une petite portion du chemin critique.

À l'inverse, le modèle AlexNet sur le système DGX-1 est dépendant du CPU à cause de la transformation d'images réalisée avant l'entraînement. Avec les données synthétiques (créées en mémoire), l'entraînement traite 20 000 images par seconde, alors que les deux sources d'E/S sont limitées à environ 9200 images/sec, soit environ 1,2 Go/s. Cette vitesse est inférieure à la vitesse de lecture mesurée sur le stockage, ce qui signifierait que le ralentissement se fait plutôt au niveau du traitement des fichiers image que des E/S du stockage externe.

Remarques sur le calibrage des performances

Le dimensionnement de la plate-forme FlashBlade dépend de 1) la complexité du modèle à entraîner, 2) la taille d'origine des données et 3) le nombre de GPU fonctionnant simultanément. Le troisième facteur est directement lié au nombre de scientifiques des données dans l'équipe qui utilisent les ressources du cluster. Chacun de ces facteurs influe sur les performances requises et sur le nombre de lames nécessaires à l'apprentissage profond et aux autres étapes du pipeline.

- Les modèles les plus complexes demandent plus de calculs pour chaque exemple d'entraînement, et donc moins de débit en lecture que des modèles plus simples.
- Les fichiers de grande taille se traduisent par un débit en lecture plus élevé pour chaque lot d'entrée depuis la couche de stockage. Notez que ce point n'a rien à voir avec la complexité du modèle ; les images sont redimensionnées avant l'entraînement.
- L'ajout de GPU permet d'augmenter la charge puisque plusieurs modèles distincts peuvent alors être entraînés simultanément ou que la capacité d'entraînement d'un même modèle peut être étendue. La rapidité accrue des nouvelles générations de GPU va également augmenter la demande de données.

> C'est avec les tailles d'entrée d'origine que les spécialistes des données peuvent le plus facilement transformer et traiter les données pour leur conférer la forme la plus utile.

Le lien entre taille des entrées et complexité du modèle est un point important. À titre d'exemple, les modèles ImageNet comme ResNet-50 ajustent les entrées au format d'une image de 225x225. Le prétraitement des images avant l'entraînement pour leur donner cette taille se traduit par une réduction du débit mais fige l'utilisation des données. Il ne sert à rien de créer des modèles plus grands pour utiliser des entrées de plus grande taille, sauf si le nombre d'entrées est également ajusté. Si le nombre de paramètres du modèle dépasse le nombre de points d'entrée, le modèle se contente de mémoriser les données d'entraînement, sans véritablement apprendre.

Pour vous donner une idée de la capacité à prévoir, le tableau suivant présente les modèles ImageNet les plus connus, très différents par leur taille et leur complexité. Le débit mesuré avec 8 GPU correspond à des tailles d'entrée d'origine de 150 ko, et la demande de débit est extrapolée pour les images de 1 Mo. Pour déterminer le nombre de systèmes DGX-1 susceptibles d'être alimentés par la plate-forme FlashBlade, nous supposons que 50 % du débit disponible va à l'entraînement et que les 50 % restants sont utilisés pour les étapes d'ingestion, de transformation et d'expérimentation.

MODÈLE IMAGENET	DÉBIT EN LECTURE DU SYSTÈME DGX-1		NOMBRE DE SYSTÈMES DGX-1 PAR FLASHBLADE	
	IMAGES DE 150 Ko, DEPUIS IMAGENET	IMAGES DE 1 Mo, EXTRAPOLATION	SYSTÈME 7 LAMES, UTILISANT DES IMAGES DE 1 Mo	SYSTÈME 15 LAMES, UTILISANT DES IMAGES DE 1 Mo
INCEPTION V3	2,09 Mo/S	2,09 Go/S	3	7
RESNET-50	224 Mo/S	1,49 Go/S	2	5
RESNET-152	99 Mo/S	660 Mo/S	5	11
ALEXNET	1204 Mo/S	8,03 Go/S	–	1
VGG16	158 Mo/S	1,05 Go/S	3	7

Les performances maximales de FlashBlade dépendent de la taille des fichiers d'entrée.

- Pour les petits fichiers (50 Ko), un système FlashBlade à sept lames peut atteindre un débit en lecture de 5 Go/s environ sur des données aléatoires, et un système à 15 lames peut dépasser les 10 Go/s.
- Pour les fichiers plus volumineux (de plus de 1 Mo), les performances en lecture dépassent légèrement 1 Go/s par lame, soit 15 Go/s sur un système de 15 lames.

Pour le modèle ResNet-50, avec des images qui ont conservé leur format d'origine (1 Mo par fichier), les performances requises sont de 1,5 Go/s ; deux entraînements concurrents demanderaient environ 3 Go/s de débit en lecture, ce qui laisse une marge pour l'ingestion et le traitement des données, le débogage du modèle et les tests.

À l'avenir, des GPU plus puissants seront commercialisés, comme le V100 qui devrait au moins multiplier par trois la vitesse d'entraînement par rapport au P100 et augmenter les performances exigées du système de stockage. Des GPU plus puissants se traduisent aussi par des réseaux de neurones plus profonds, qui augmentent le volume de calcul par lot d'entrée. L'augmentation des performances nécessaire sera plus ou moins compensée par l'existence de modèles plus profonds.

Alternative : placer des données sur un stockage local (SSD)

Pour les petits jeux de données, le stockage partagé haute performance peut être remplacé par des SSD locaux (8 To au total), sur le DGX-1, où sont placées les données depuis un niveau de stockage plus lent. À chaque nouveau jeu d'entraînement, l'utilisateur doit déployer les nouvelles données. Cette méthode n'est pas envisageable pour les jeux de données dont le volume dépasse la capacité du stockage local.

Le débit en écriture mesuré sur les quatre SSD RAID-0 est de 360 Mo/s pour un flux unique et de 680 Mo/s pour plusieurs flux.
En extrapolant, on obtient les temps de chargement local des données présentés dans le tableau ci-après. La copie à flux unique correspond à « cp -r ».

	FLUX UNIQUE	FLUX MULTIPLES
1 To	48,5 MINUTES	25,7 MINUTES
8 To	6,5 HEURES	3,4 HEURES

Si, par exemple, l'entraînement utilise un jeu d'entrée de 8 To, le DGX-1 doit attendre environ 3,4 heures que les nouvelles données soient chargées sur les disques SSD. Pendant ce temps, les GPU du système DGX-1 sont inoccupés. Si l'on mesure le temps total jusqu'à l'obtention d'une solution, le stockage partagé assure des performances nettement plus rapides.

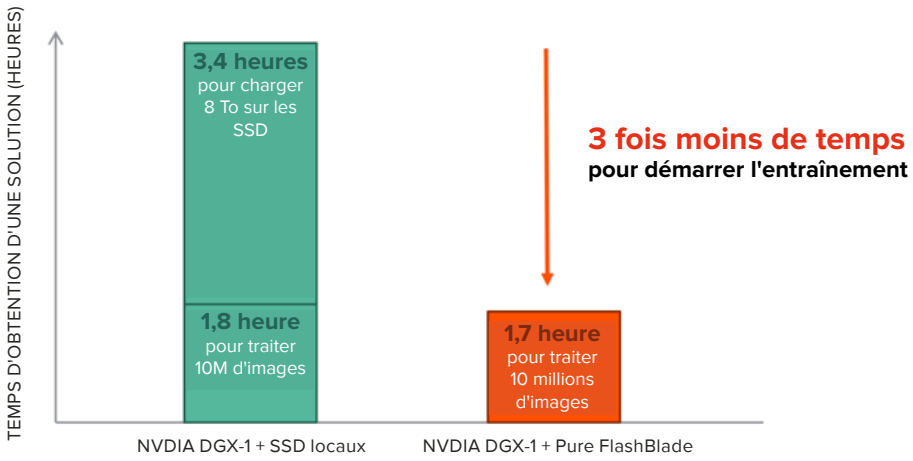


FIGURE 7 : Test de performance sous TensorFlow pour le modèle ResNet-50, avec mesure du temps d'entraînement pour 10 millions d'images incluant le temps de transfert des données sur les SSD.

ANNEXE : ARCHITECTURE D'UN SYSTÈME EN PRODUCTION

L'architecture du système complet doit être capable de stocker les importants jeux de données nécessaires à l'apprentissage profond et de les relier aux GPU chargés de l'entraînement des modèles. Le schéma suivant illustre l'architecture réelle d'un système d'apprentissage profond effectivement déployé pour des applications de conduite autonome.

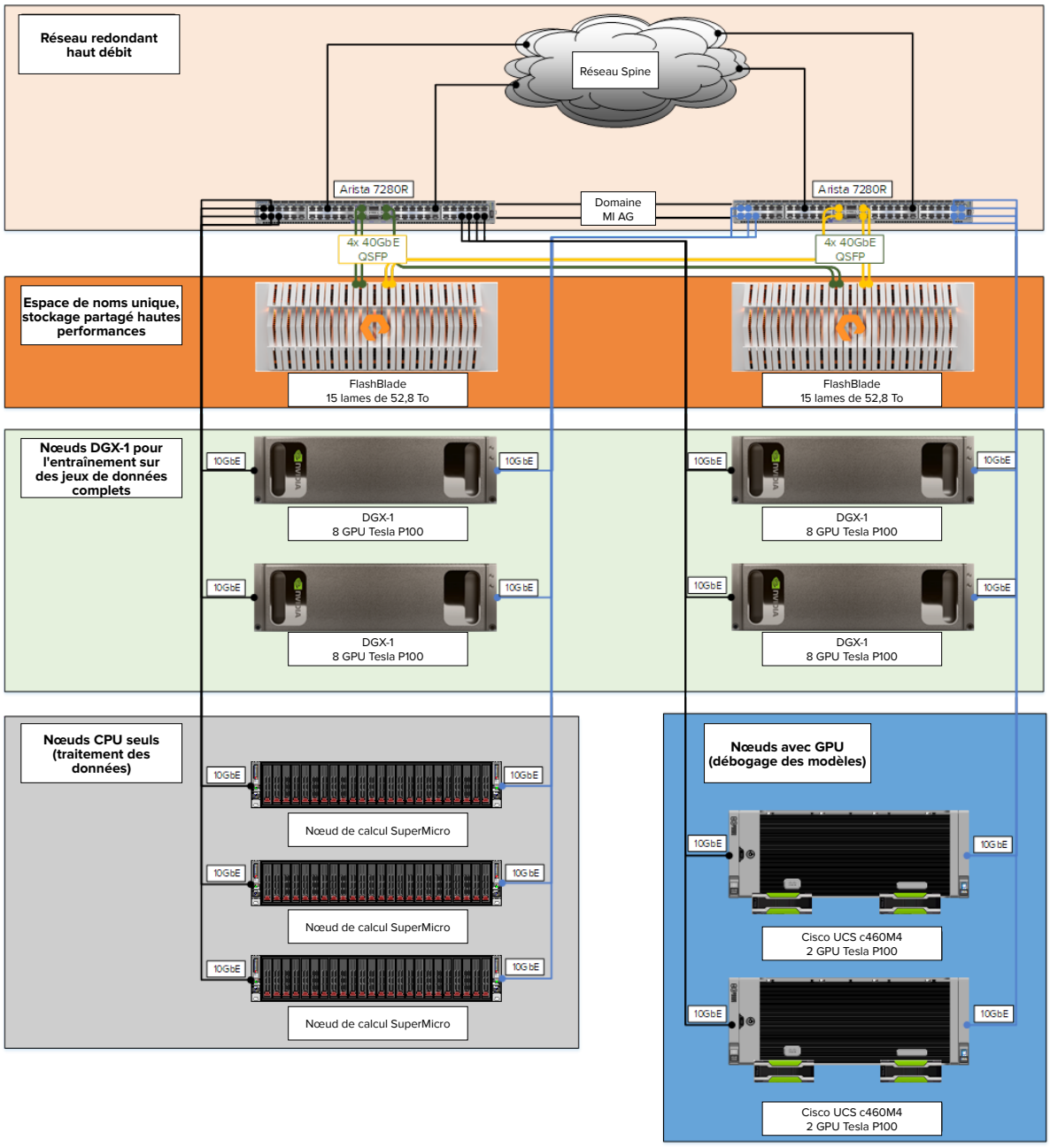


FIGURE 8 : Exemple d'architecture d'un système complet

Infrastructure physique

L'architecture de la Figure 8 inclut les éléments suivants pour l'infrastructure complète.

Capacité de calcul principale : DGX-1 (3U)

- 8 GPU Nvidia P100 (à l'avenir : GPU V100)
- Connectivité directe L2 de FlashBlade au système DGX-1 via une liaison Ethernet
- Réseau configuré sur le même sous-réseau que les VIP FlashBlade

Stockage : FlashBlade (4U)

- 7 à 15 lames selon les performances souhaitées
- Lames de 17 To ou 52 To selon la taille des jeux de données
- Les lames de 17 To ont des performances trois fois plus élevées que les lames de 52 To si l'on considère le ratio performance/capacité
- Configuration incluant plusieurs VIP de données et un seul système de fichiers

Serveurs de calcul supplémentaires

- Pipeline d'ingestion des données qui prétraitre, nettoie et normalise les nouvelles données
- Validation à petite échelle avec modèles expérimentaux de réseaux neuronaux
- En option, serveurs équipés de GPU pour augmenter les performances de débogage du modèle d'entraînement sans prendre du temps sur un système DGX-1

Gestion du réseau : Paire de switches ToR dans une configuration MLAG

- Chaque DGX-1 a son propre canal Port Channel, chaque interface est connectée à un switch ToR pour plus de fiabilité
- Exemple de câblage pour deux serveurs DGX-1 (DGX-1-1 et DGX-1-2) :
  - DGX-1-1.port1 -> switch1
  - DGX-1-1.port2 -> switch2
  - DGX-1-2.port1-> switch1
  - DGX-1-2.port2-> switch2
- FlashBlade dans son propre canal Port Channel connecté aux deux switches ToR
  - Connectivité : FM1->switch1, FM1->switch2, FM2->switch1, FM2->switch2

Configuration de FlashBlade

Un seul système de fichiers pour l'ensemble des données d'entraînement

LISTE PUREUSER@SN1-FB-E02-33-1:~$ PUREFS							
NOM	TAILLE	UTILISÉ	% UTILISÉ	CRÉÉ	PROTOCOLES	RÈGLES	SUPPRESSION RAPIDE
NFS0	1T	138.40G	14%	2017-08-03 14:08:04 PDT	NFS	*(RW,NO_ROOT_SQUASH)	FALSE

VIP multiples pour l'accès à FlashBlade

LISTE PUREUSER@SN1-FB-E02-33-1:~$ PURENETWORK							
NOM	ACTIVÉ	SOUS-RÉSEAU	ADRESSE	MASQUE VLAN	PASSERELLE	MTU	SERVICES
NFS	TRUE	NET3	10.21.115.9	2115	10.21.115.1	1500	DONNÉES
NFS0	TRUE	NET3	10.21.115.6	2115	10.21.115.1	1500	DONNÉES
NFS1	TRUE	NET3	10.21.115.7	2115	10.21.115.1	1500	DONNÉES
NFS2	TRUE	NET3	10.21.115.8	2115	10.21.115.1	1500	DONNÉES
VIR0	TRUE	NET2	10.21.112.113	2112	10.21.112.1	1500	GESTION

Configuration du système DGX-1

Montages pour FlashBlade

```
pureuser@sn1-dgx-1-e02-37:~/tensorflow-benchmarks$ cat /etc/mtab
/dev/sda2 / ext4 rw,errors=remount-ro 0 0
proc /proc proc rw,noexec,nosuid,nodev 0 0
sysfs /sys sysfs rw,noexec,nosuid,nodev 0 0
.....
/dev/sda1 /boot/efi vfat rw 0 0
/dev/sdb1 /raid ext4 rw 0 0
rpc_pipefs /run/rpc_pipefs rpc_pipefs rw 0 0
systemd /sys/fs/cgroup/systemd cgroup rw,noexec,nosuid,nodev,none,name=systemd 0 0
10.21.115.6:/nfs0 /mnt/nfs0 nfs rw,addr=10.21.115.6,_netdev 0 0
10.21.115.7:/nfs0 /mnt/nfs1 nfs rw,addr=10.21.115.7,_netdev 0 0
10.21.115.8:/nfs0 /mnt/nfs2 nfs rw,addr=10.21.115.8,_netdev 0 0
10.21.115.9:/nfs0 /mnt/nfs3 nfs rw,addr=10.21.115.9,_netdev 0 0
```

Table de routage pour répartir le trafic du point de montage entre les deux interfaces

```
pureuser@sn1-dgx-1-e02-37:~/tensorflow-benchmarks$ ip route
default via 10.21.115.1 dev em1
10.21.115.0/24 dev em1  proto kernel  scope link  src 10.21.115.120
10.21.115.6 dev em1  proto kernel  scope link  src 10.21.115.120
10.21.115.7 dev em2  proto kernel  scope link  src 10.21.115.113
10.21.115.8 dev em1  proto kernel  scope link  src 10.21.115.120
10.21.115.9 dev em2  proto kernel  scope link  src 10.21.115.113
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1
```

Captures d'écran prises pendant l'entraînement

```
Every 2.0s: nvidia-smi

Sat Aug 19 13:00:13 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375.66                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-SXM2...  On   | 0000:06:00.0     Off |                    0 |
| N/A   49C    P0   190W / 300W |  15661MiB / 16276MiB |     94%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P100-SXM2...  On   | 0000:07:00.0     Off |                    0 |
| N/A   44C    P0   210W / 300W |  15661MiB / 16276MiB |     90%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla P100-SXM2...  On   | 0000:0A:00.0     Off |                    0 |
| N/A   48C    P0   224W / 300W |  15661MiB / 16276MiB |     83%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla P100-SXM2...  On   | 0000:0B:00.0     Off |                    0 |
| N/A   48C    P0   221W / 300W |  15661MiB / 16276MiB |     87%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla P100-SXM2...  On   | 0000:85:00.0     Off |                    0 |
| N/A   50C    P0   208W / 300W |  15661MiB / 16276MiB |     94%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla P100-SXM2...  On   | 0000:86:00.0     Off |                    0 |
| N/A   50C    P0   198W / 300W |  15661MiB / 16276MiB |     92%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla P100-SXM2...  On   | 0000:89:00.0     Off |                    0 |
| N/A   48C    P0   209W / 300W |  15661MiB / 16276MiB |     95%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla P100-SXM2...  On   | 0000:8A:00.0     Off |                    0 |
| N/A   47C    P0   219W / 300W |  15661MiB / 16276MiB |     86%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     48060    C   python                                     15655MiB |
|    1     48060    C   python                                     15655MiB |
|    2     48060    C   python                                     15655MiB |
|    3     48060    C   python                                     15655MiB |
|    4     48060    C   python                                     15655MiB |
|    5     48060    C   python                                     15655MiB |
|    6     48060    C   python                                     15655MiB |
|    7     48060    C   python                                     15655MiB |
+-----------------------------------------------------------------------------+
```

FIGURE 9 : Flux des données d'entraînement dans l'architecture logicielle

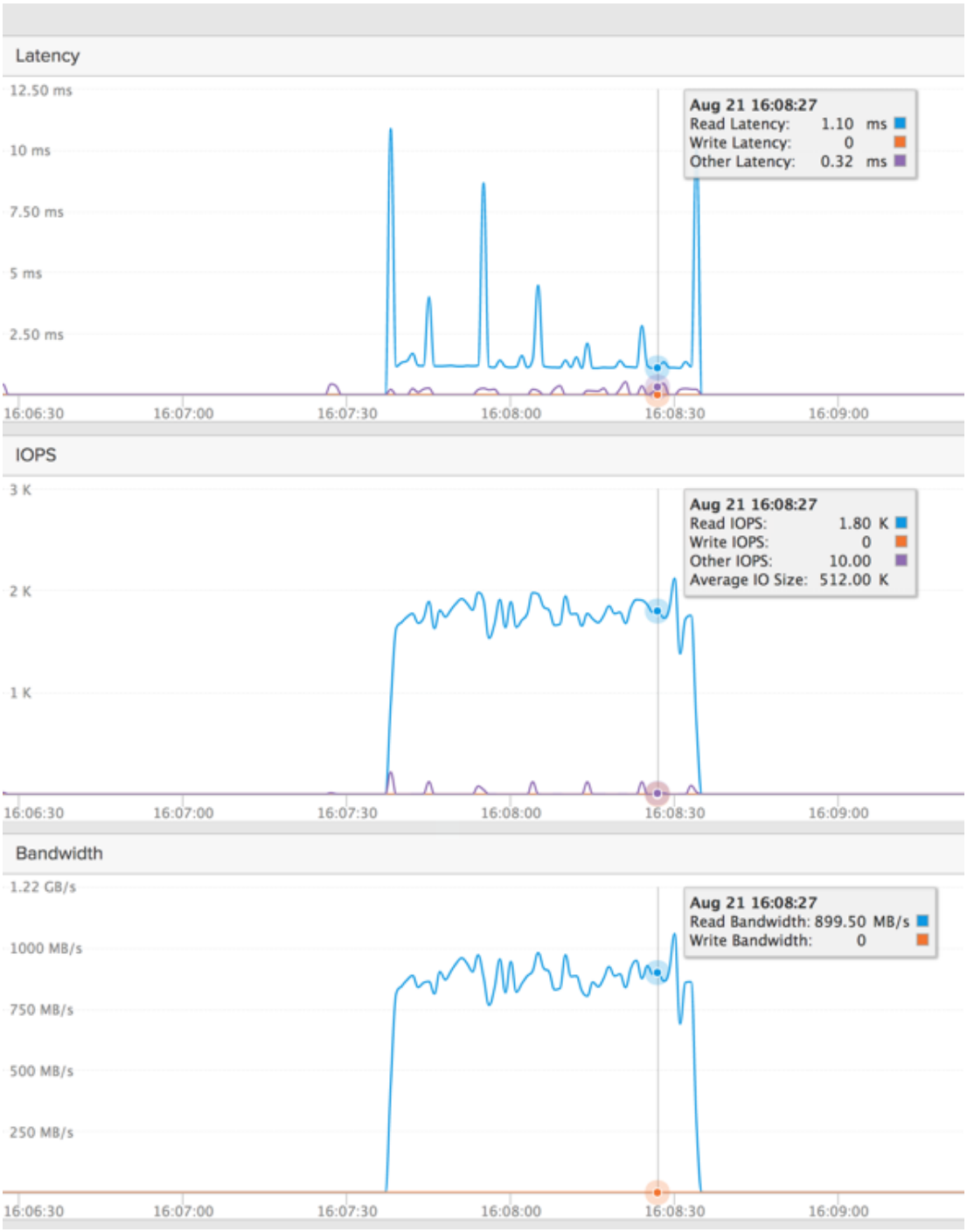


FIGURE 10 : Flux des données d'entraînement dans l'architecture logicielle

ps_wp21p_AI-reference-architecture_ltr_01