OPTIMISEZ L'EFFICACITÉ DU GPU AVEC LES PROCESSEURS HAUTE FRÉQUENCE AMD EPYC™

LIVRE BLANC | 2025

Dans le monde de l'IA et de l'apprentissage automatique, l'exécution de charges de travail d'entraînement et d'inférence à grande échelle nécessite une grande quantité de ressources de calcul. Pour de nombreuses charges de travail, les GPU sont devenus essentiels dans le développement et le déploiement de grands modèles d'IA. Les opérateurs de centres de données de toutes tailles et échelles doivent s'assurer que leurs investissements dans des GPU leur apportent une valeur maximale. L'augmentation de l'utilisation et de l'efficacité des GPU doit être une priorité pour toute personne prenant en charge ou utilisant des charges de travail d'IA basées sur GPU.

Les GPU sont au centre des préoccupations pour les applications d'IA qui traitent les lourdes charges que représentent l'entraînement et l'inférence, mais les CPU jouent un rôle essentiel dans la coordination de la charge de travail globale et exercent un impact significatif sur l'efficacité des GPU. Les CPU hôtes effectuent des tâches cruciales telles que le pré-traitement et le post-traitement des données, ainsi que la gestion de leur déplacement. Les demandes d'inférence utilisant un grand modèle de langage (LLM) sont généralement exécutées par un système basé sur GPU, contenant un CPU hôte qui gère les requêtes entrantes ou les demandes de modèle, également appelées invites. Parmi les exemples les plus courants d'applications utilisant l'inférence LLM figurent l'analyse des sentiments, la traduction, la création de contenu, la synthèse et les chatbots question-réponse. Chacune de ces applications a des structures différentes d'invite et de réponse. La fonction principale d'un CPU hôte consiste à coordonner le traitement de chaque requête pour réduire le temps d'inactivité sur les GPU.

Dans cet article, nous allons découvrir comment le CPU hôte d'un système basé sur GPU peut améliorer significativement les performances globales et la rentabilité de l'inférence LLM. Nous allons nous concentrer sur les fonctions clés du CPU pendant l'inférence, montrer comment les performances du CPU hôte peuvent réduire le temps de latence de bout en bout et présenter des performances jusqu'à 24 % supérieures, avec un temps de latence moyen amélioré de 9 % grâce à AMD Instinct MI300 et de 8 % grâce à NVIDIA H100, respectivement, en utilisant les CPU haute fréquence AMD EPYC™ en tant que processeur hôte.

Implication du CPU hôte dans l'inférence LLM

Dans le monde de l'IA et de l'apprentissage automatique, l'exécution de charges de travail d'entraînement et d'inférence à grande échelle nécessite une grande quantité de ressources de calcul. Pour de nombreuses charges de travail, les GPU sont devenus essentiels dans le développement et le déploiement de grands modèles d'IA. Les opérateurs de centres de données de toutes tailles

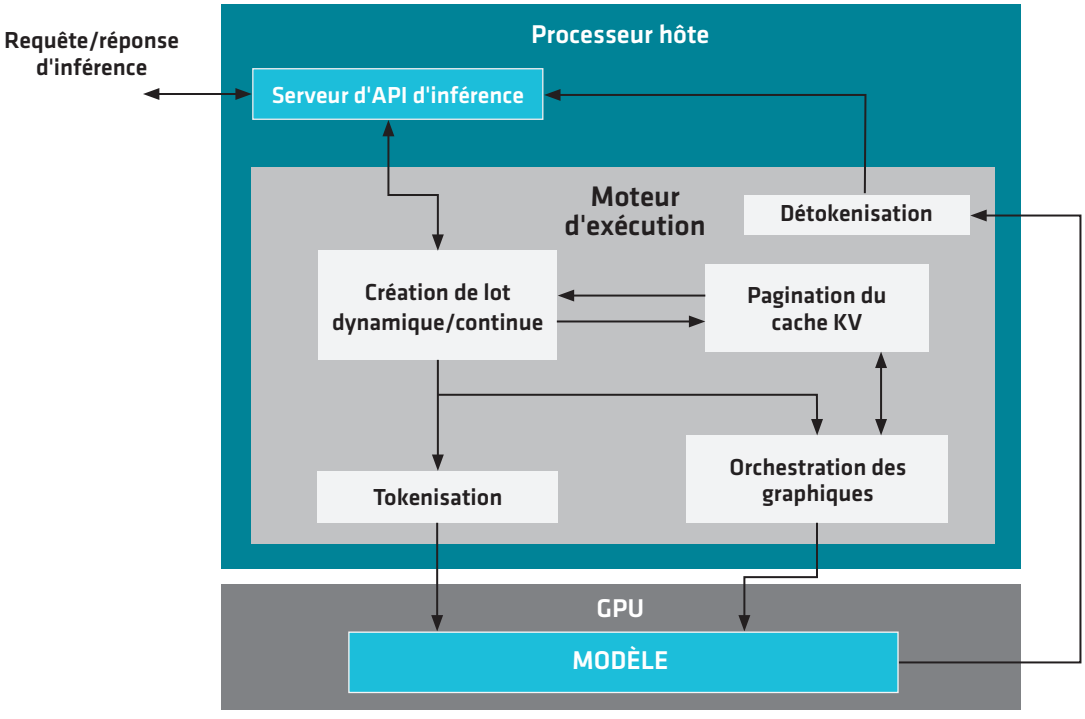

Figure 1 : Bloc fonctionnel fondamental du CPU de l'hôte généralisé

et échelles doivent s'assurer que leurs investissements dans des GPU leur apportent une valeur maximale. L'augmentation de l'utilisation et de l'efficacité des GPU doit être une priorité pour toute personne utilisant ou prenant en charge des charges de travail IA basées sur GPU. La Figure 1 illustre le flux fondamental d'une requête d'inférence dans un système basé sur GPU avec le modèle (illustré en bleu)

exécuté sur le GPU et les autres fonctions exécutées sur le CPU hôte. La section suivante décrit les fonctions fondamentales et facultatives utilisées pour fournir un service d'inférence.

Serveur d'API d'inférence :
Le serveur d'API d'inférence traite une requête entrante et la transfère au moteur d'exécution. Le moteur d'exécution génère une réponse à la fin de la tâche qui est renvoyée au demandeur. Les actions de l'API d'inférence peuvent être en grande partie découplées du moteur d'exécution à l'aide de files d'attente au niveau de l'interface entre les deux. Il s'agit d'une fonction particulièrement importante lorsqu'il existe plusieurs utilisateurs simultanés, c'est-à-dire plusieurs invites, plusieurs modèles simultanés et/ou plusieurs GPU fonctionnant avec un CPU hôte partagé. L'API garantit que la réponse à une invite est renvoyée au demandeur approprié.

Moteur d'exécution :
Le moteur d'exécution est représenté par les fonctions dans la case orange de la Figure 1. Le moteur d'exécution au sein du CPU exécute des fonctions de gestion des ressources critiques, telles que le traitement par lots dynamique et la pagination du cache K-V, afin de garantir l'optimisation de l'efficacité du calcul et de l'utilisation de la mémoire du GPU. Il doit également gérer les tâches d'orchestration telles que le lancement du noyau et la synchronisation sur plusieurs GPU. Ces tâches peuvent se trouver sur le chemin critique et avoir un impact direct sur le temps de latence de bout en bout des requêtes d'inférence. De plus, à mesure que les architectures de modèles évoluent pour inclure des éléments tels que le flux de contrôle dépendant des données, le temps de réponse du CPU sera plus long.

Exécution du brouillon de modèle (facultatif) :
Pour certaines applications, un brouillon de modèle peut être utilisé pour spéculer avant d'exécuter l'inférence complète et de faire des prévisions précoces. Ces prévisions précoces équivalent à la réduction de l'espace d'état et permettent au modèle principal d'exécuter des tailles de lots plus élevées à l'aide du LLM principal pendant la génération, ainsi que de réduire le nombre d'étapes de génération de token. Dans ce mode de fonctionnement, le brouillon de modèle peut être exécuté sur le CPU hôte pour permettre de dédier les ressources de calcul et de mémoire du GPU au LLM principal.

Prétraitement :
Pour certaines applications, le pré-traitement implique l'exécution de modèles plus petits, tels que Sentence-BERT, afin de générer des intégrations pour les systèmes de récupération, ou « ingénierie d'invites ». Une fois prétraitées, les invites d'un lot sont tokenisées et préparées pour l'exécution sur le ou les GPU pendant la phase d'inférence principale. L'exécution du modèle d'intégration et la recherche de similarité de la base de données vectorielle ultérieure par le récupérateur peuvent se trouver sur le chemin critique d'inférence. Un CPU hôte à hautes performances peut diminuer les coûts de cette action.

Post-traitement :
Une fois que le GPU a terminé l'exécution du modèle, le CPU finalise la réponse en gérant l'échantillonnage de token et en effectuant d'autres tâches de traitement de sortie telles que le formatage, la gestion des erreurs ou la visualisation des données de réponse, pour présenter l'utilisateur.

Opérations de ML :
Dans un environnement de production où une requête d'inférence fait un choix entre plusieurs modèles (voir https://arxiv.org/pdf/2405.07518), le temps de chargement du modèle est particulièrement important. Un CPU doté d'un bon IPC, d'une bonne mémoire et d'une bonne bande passante d'E/S peut considérablement améliorer les performances de chargement du modèle.

Chacune de ces fonctions que le CPU hôte exécute peut avoir un impact significatif sur l'efficacité du ou des GPU effectuant l'inférence, et donc sur le temps de réponse total pour une requête d'inférence.

Rôle des CPU haute fréquence

Bien que certaines des actions répertoriées dans la section précédente puissent s'exécuter en même temps que l'exécution de l'inférence du GPU, d'autres peuvent se trouver sur le chemin critique pour le temps de latence, comme le lancement du noyau, la tokenisation, le groupage dynamique, l'exécution du brouillon de modèle, le flux de contrôle dépendant des données, la synchronisation, etc. Lorsque l'exécution a lieu dans le chemin critique de latence, l'efficacité et la vitesse auxquelles ces fonctions exécutent leur tâche peuvent devenir un facteur important dans le temps de réponse d'inférence total. Ces actions du CPU hôte deviennent plus critiques lorsque les temps d'exécution d'inférence sont contraints par le temps de latence.

Nos précédentes études internes ont inclus des mesures approfondies de l'activité du CPU hôte pour les charges de travail d'inférence et d'entraînement sur les systèmes basés sur NVIDIA H100 et AMD Instinct™ MI300. Nous avons rassemblé ces profils en utilisant TRT-LLM (système H100) et vLLM pour l'inférence. Pour l'entraînement, nous avons rassemblé des profils utilisant des structures JAX, PyTorch et Megratron-LM.

Ces profils nous ont montré que les performances à thread unique du CPU hôte étaient plus critiques que le débit pour atténuer les frais supplémentaires potentiels liés à l'activité du CPU hôte.
AMD propose ainsi une série de processeurs haute fréquence avec 16 à 64 cœurs, ce qui permet aux utilisateurs de répondre efficacement aux exigences du processeur hôte tout en contrôlant les spécifications et les coûts du système. Pour les systèmes GPU volumineux, les partenaires de l'écosystème AMD et d'IA recommandent un système basé sur AMD EPYC™ 9575F avec 64 cœurs, avec une plage de TDP de 320 à 400 watts, et avec une fréquence maximale du cœur (Fmax) de 5 GHz pour améliorer les performances globales d'inférence de bout en bout.

Sensibilité de l'hôte - Configuration du test et résultats

Nous avons mené une étude pour démontrer l'impact des performances du CPU hôte sur les performances globales du système, ou « sensibilité de l'hôte ». Nous présentons les détails de l'expérience réalisée pour mettre en évidence les avantages d'AMD EPYC™ 9575F en tant que CPU hôte dans un système basé sur GPU. Notre étude portait principalement sur la mesure du temps de latence de bout en bout de l'inférence à l'aide de combinaisons de longueur d'invite et de longueur de sortie représentatives des tâches d'inférence de chatbot, de création de contenu, de synthèse et de traduction. Nous avons choisi des tailles de lots de 32 et 1 024 pour représenter l'inférence en ligne et hors ligne, respectivement.

Nous n'avons pas mis en œuvre un serveur d'inférence, un pipeline RAG, l'inférence multi-modèles COE ni le modèle préliminaire pour faire des spéculations dans cette étude. Cette étude a été réalisée à l'aide de points de contrôle FP8 de modèles open source populaires pour l'inférence.

TABLEAU 1 : CONFIGURATION DE L'EXPÉRIENCE DE LA SENSIBILITÉ DE L'HÔTE

EXÉCUTION : VLLM : 0.7	MODÈLES :	CPU HÔTE :	SYSTÈMES :

• Création de lot continue activée
• Num-scheduler-steps = 1
• (longueur d'invite, tokens de sortie)
  • Chatbot = (128,128)
  • Création de contenu = (128,1 024/2 048)
  • Synthèse = (1 024/2 048, 128)
  • Traduction = (1 024/2 048, 1 024/2 048)
• Tailles de lots = [32, 1 024]
• Tokenisation = hors ligne
• Détokenisation = en ligne
	• Llama3.1-70B-Instruct (FP8) ; parallélisme (tenseur) = 8
	• Llama3.1-8B-Instruct (FP8) ; parallélisme (tenseur) = 1
	• Mixtral 8x7B-Instruct (FP8) ; parallélisme (tenseur) = 8
		• AMD EPYC™ 9575F avec TDP de 320-400 W ; Fmax = 5 GHz
		• Intel Xeon 8592+ avec TDP de 350 W ; Fmax = 3,9 GHz
			• 8x AMD Instinct™ MI300 - Appareil 74a1-XGMI-192GB-750W ; ROCm™ 6.3.0-39
			  • Système d'exploitation hôte : 575F – UBUNTU 24.04 LTS ; 8592+ 24.04.1 LTS
			• 8x H100 NVIDIA H100-80Go-HBM3-700W ; Cuda version 12.6
			  • Système d'exploitation hôte : 9575F- Ubuntu 22.04.4 LTS ; 8592+ - Ubuntu 22.04.5 LTS

Sensibilité de l'hôte - Résultats

Les tableaux suivants présentent les avantages en matière de performances d'un CPU hôte EPYC 9575F par rapport au Xeon 8592+ dans les systèmes de GPU basés sur 8x AMD Instinct™ MI300x et sur 8x NVIDIA H100, sur différentes tâches d'inférence. Pour chaque test, nous avons effectué trois exécutions et recueilli des mesures. La médiane de ces mesures est indiquée dans les tableaux suivants.

TABLEAU 2 : COMPARAISON DES HÔTES : SYSTÈME BASÉ SUR GPU 8x AMD INSTINCT™ MI300x

MODÈLE	TÂCHE	TAILLE DU LOT	TOKENS D'ENTRÉE	TOKENS DE SORTIE	AMÉLIORATION DU TEMPS DE LATENCE AMD EPYC™ 9575F/XEON 8592+
Llama-3.1-8B-Instruct-FP8	Chatbot	32	128	128	x 1,06
		1024	128	128	x 1,05
	Création de contenu	32	128	1024	x 1,07
		1024	128	1024	x 1,03
	Synthèse	32	1024	128	x 1,05
		1024	1024	128	x 1,03
	Traduction	32	128	1024	x 1,05
		1024	1024	1024	x 1,03

TABLEAU 2 - SUITE : COMPARAISON DES HÔTES : SYSTÈME BASÉ SUR GPU 8x AMD INSTINCT™ MI300x

MODÈLE	TÂCHE	TAILLE DU LOT	TOKENS D'ENTRÉE	TOKENS DE SORTIE	AMÉLIORATION DU TEMPS DE LATENCE AMD EPYC™ 9575F/XEON 8592+
Llama-3.1-70B-Instruct-FP8	Chatbot	32	128	128	x 1,13
		1024	128	128	x 1,08
	Création de contenu	32	128	1024	x 1,10
		1024	128	1024	x 1,05
	Synthèse	32	1024	128	x 1,08
		1024	1024	128	x 1,03
	Traduction	32	1024	1024	x 1,10
		1024	1024	1024	x 1,05
Mixtral 8x7B-Instruct-FP8	Chatbot	32	128	128	x 1,19
		1024	128	128	x 1,15
	Création de contenu	32	128	1024	x 1,13
		1024	128	1024	x 1,16
	Synthèse	32	1024	128	x 1,24
		1024	1024	128	x 1,11
	Traduction	32	1024	1024	x 1,11
		1024	1024	1024	x 1,15

TABLEAU 3 : SYSTÈME BASÉ SUR UN GPU NVIDIA H100 8X POUR COMPARAISON D'HÔTES

MODÈLE	TÂCHE	TAILLE DU LOT	TOKENS D'ENTRÉE	TOKENS DE SORTIE	AMÉLIORATION DU TEMPS DE LATENCE AMD EPYC™ 9575F/XEON 8592+
Llama-3.1-8B-Instruct-FP8	Chatbot	32	128	128	x 1,12
		1024	128	128	x 1,18
	Création de contenu	32	128	2048	x 1,09
		1024	128	2048	x 1,09
	Synthèse	32	2048	128	x 1,05
		1024	2048	128	x 1,04
	Traduction	32	2048	2048	x 1,05
		1024	2048	2048	x 1,04
Llama-3.1-70B-Instruct-FP8	Chatbot	32	128	128	x 1,04
		1024	128	128	x 1,11
	Création de contenu	32	128	2048	x 1,04
		1024	128	2048	x 1,09
	Synthèse	32	2048	128	x 1,01
		1024	2048	128	x 1,03
	Traduction	32	2048	2048	x 1,02
		1024	2048	2048	x 1,05
Mixtral 8x7B-Instruct-FP8	Chatbot	32	128	128	x 1,09
		1024	128	128	x 1,20
	Création de contenu	32	128	2048	x 1,10
		1024	128	2048	x 1,19
	Synthèse	32	2048	128	x 1,06
		1024	2048	128	x 1,05
	Traduction	32	2048	2048	x 1,08
		1024	2048	2048	x 1,13

CONCLUSION

Les tableaux ci-dessus montrent clairement que le temps de latence d'inférence est impacté par le choix du CPU hôte. La fréquence supérieure et les performances monothread exceptionnelles d'AMD EPYC™ 9575F permettent de réduire les coûts du CPU associé à l'exécution d'une tâche d'inférence hors ligne ou en ligne.

Sur les trois modèles représentatifs et les quatre tâches d'inférence de notre expérience, AMD EPYC™ 9575F offre une amélioration moyenne des performances de 9 % par rapport aux systèmes basés sur le GPU avec 8x AMD Instinct™ MI300, et de 8 % par rapport aux systèmes basés sur le GPU avec 8x NVIDIA H100. Pour le cas d'utilisation du chatbot, une partie importante du temps de latence de bout en bout est consacrée à la configuration des lots et à la gestion de la mémoire du GPU. Pour les cas impliquant de nombreuses étapes de génération de tokens, comme la création de contenu et la traduction, la détokenisation a été un facteur significatif du temps de latence global.

Nos futurs travaux devraient inclure une répartition détaillée de l'activité de l'hôte dans ces cas d'inférence. Nous étudierons également la sensibilité de l'hôte pour une charge de travail de traitement d'inférence avec des contraintes de temps de latence.

AUTEURS :

Ram Sivaramakrishnan : Ingénieur en conception de systèmes, architecte IA pour les solutions de serveur.
ram.sivaramakrishnan@amd.com

Matt Ouellette : Directeur de l'ingénierie du développement de produits, gestion des produits AIG-AI
matt.ouelette@amd.com

Ajith Sirra : Ingénieur en application de produit MTS, gestion des produits AIG-AI.
ajith.sirra@amd.com

Shubin Zhao : Ingénieur en conception de systèmes software SMTS, ingénieur des performances DCGPU
shubin.zhao@amd.com

Danyang Zhang : Ingénieure en développement software SMTS, gestion des produits AIG-AI
danyang.zhang@amd.com

Jeremy Arnold : Ingénieur en conception de systèmes software PMTS, ingénieur des performances DCGPU
jeremy.arnold@amd.com

Mary Cirino : Ingénieure en développement software MTS, ingénieur des performances DCGPU
mary.cirino@amd.com