Red Hat

Bien débuter avec l'inférence d'IA

Accédez plus rapidement à l'efficacité

Sommaire

Introduction 3

Termes clés 4

L'évolution des grands modèles de langage 7

Les défis liés au déploiement de l'inférence 9

Une approche globale pour de meilleures performances d'inférence 10

Une double approche pour des modèles plus efficaces 12

1. Optimisation de l'exécution de l'inférence (vLLM) 12

2. Optimisation du modèle d'IA 14

Red Hat AI 18

Présentation de Red Hat AI 18

Optimisation des modèles avec Red Hat 20

Étapes suivantes 22

Introduction

L'optimisation de l'inférence des modèles d'IA est l'une des méthodes les plus efficaces pour réduire les coûts d'infrastructure, diminuer la latence et améliorer le débit, en particulier dans le cadre du déploiement de grands modèles en production.

Ce livre numérique développe les bases pour comprendre l'ingénierie des performances de l'inférence ainsi que l'optimisation des modèles. Il présente notamment la quantification, l'élagage et d'autres techniques qui limitent les besoins en matière de calcul et de mémoire, ainsi que des systèmes d'exécution tels que vLLM (Virtual Large Language Model) qui améliorent l'efficacité de l'inférence.

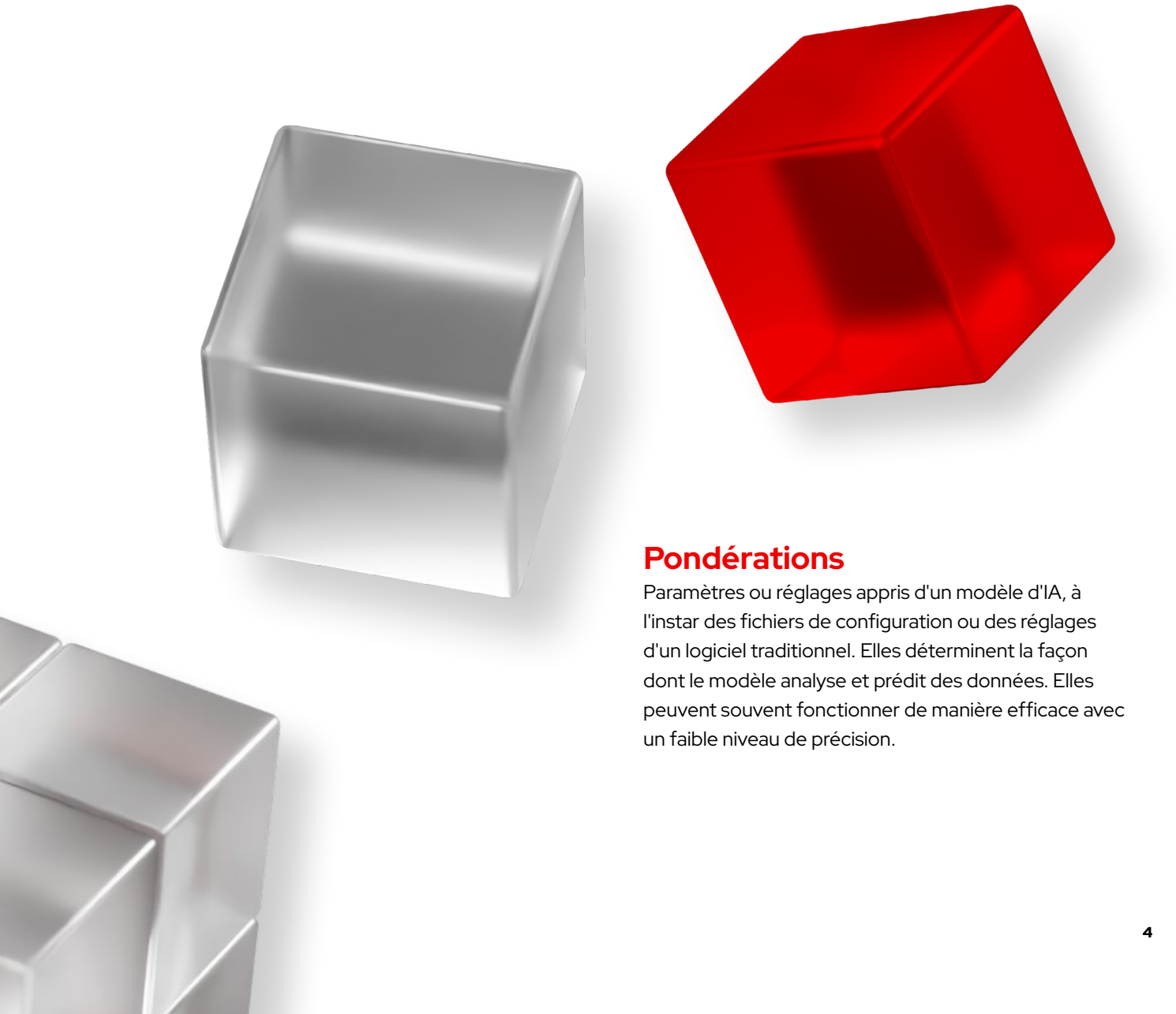
Il met également en avant les avantages de l'approche ouverte de Red Hat, du référentiel de modèles validés et des outils tels que LLM Compressor et Red Hat® AI Inference Server. Que vous utilisiez des processeurs graphiques (GPU), des unités de traitement de tenseur (TPU) ou d'autres accélérateurs, ce guide vous fournira des informations pratiques pour créer des systèmes d'inférence d'IA plus intelligents et plus efficaces.

Termes clés

Comprendre les composants d'un modèle

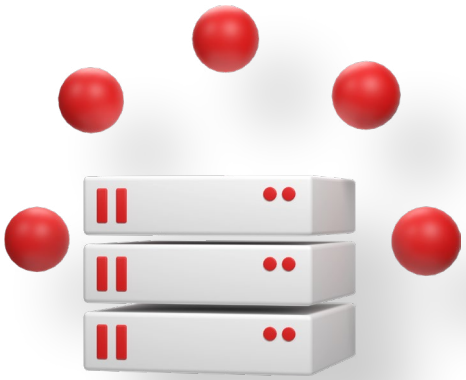
Activations

Données temporaires générées lorsqu'un modèle traite des informations (jetons textuels d'entrée), à l'image des résultats intermédiaires produits lors d'un calcul. Elles nécessitent généralement un haut niveau de précision pour fournir des résultats exacts.



Pondérations

Paramètres ou réglages appris d'un modèle d'IA, à l'instar des fichiers de configuration ou des réglages d'un logiciel traditionnel. Elles déterminent la façon dont le modèle analyse et prédit des données. Elles peuvent souvent fonctionner de manière efficace avec un faible niveau de précision.



Quantification

La quantification est une technique qui réduit la taille et les besoins en ressources des modèles d'IA en stockant leurs paramètres (pondérations) et données intermédiaires (activations) dans des formats moins précis qui utilisent un nombre inférieur de bits par valeur. Grâce à cette technique, il est possible de gérer les ressources plus efficacement, à l'instar de la compression des fichiers sur un ordinateur. Les **performances du modèle sont préservées** si la quantification est correctement réalisée.

- La **quantification des pondérations** réduit la taille de stockage des paramètres du modèle et **optimise l'utilisation de la mémoire pendant l'inférence**[1].
- La **quantification des activations** réduit les besoins en mémoire des résultats intermédiaires (données temporaires) pendant l'inférence et permet ainsi de bénéficier d'une **exécution plus rapide et efficace**[2].
- La **quantification du cache clé-valeur** réduit l'empreinte mémoire des tenseurs clé-valeur mis en cache, ce qui permet aux modèles de **gérer plus efficacement les requêtes longues et simultanées**[3].

Niveaux de précision pour la quantification sur 16 bits, 8 bits et 4 bits :

- **Le format standard 16 bits (FP16/BF16)** préserve la précision, mais utilise beaucoup de mémoire et s'avère coûteux pour les très grands modèles.
- **Le format 8 bits (FP8/INT8)** utilise environ deux fois moins de mémoire que le format 16 bits, ce qui améliore significativement l'efficacité tout en préservant la précision du modèle.
- **Le format 4 bits (INT4)** réduit considérablement la taille du modèle et les besoins en mémoire, permettant ainsi un déploiement sur moins de ressources. L'application rigoureuse de méthodes de quantification avancées est en revanche nécessaire pour éviter toute dégradation significative de la précision.

1 Maxime Laboone, « **Introduction to Weight Quantization** », *towards data science*, 7 juillet 2023.

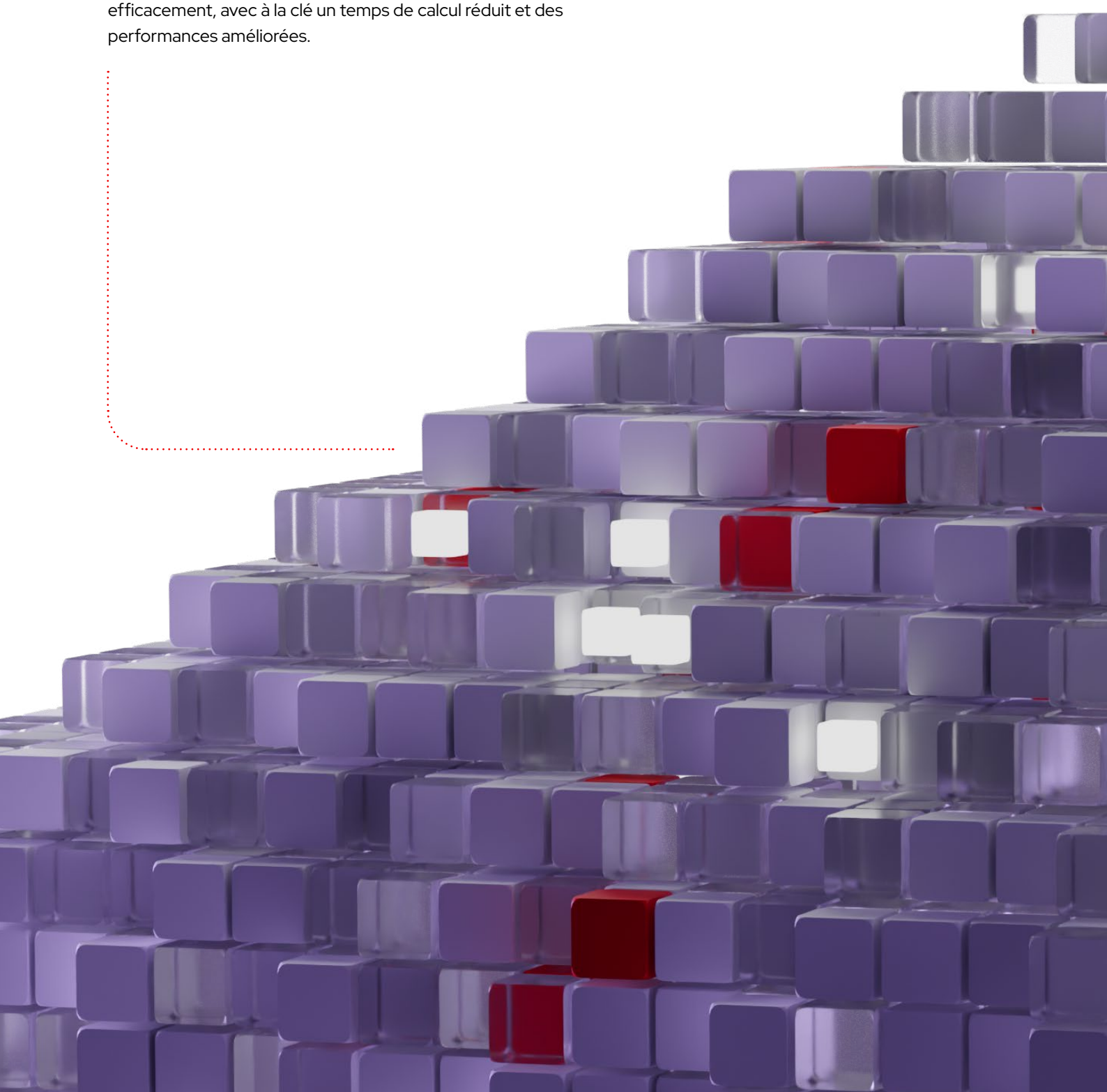
2 « **AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration** », GitHub, consulté le 8 août 2025.

3 Raushan Turganbay, « **Unlocking Longer Generation with Key-Value Cache Quantization** », *Hugging Face*, 16 mai 2024.

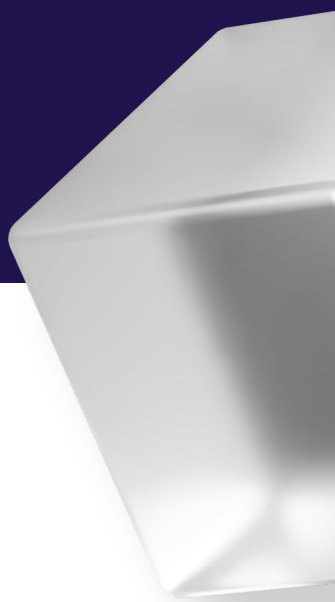
Réduire la charge de calcul grâce à l'élagage

L'**élagage** permet de réduire la demande de ressources de calcul en attribuant la valeur 0 à certains paramètres du modèle. Les systèmes évitent ainsi d'effectuer des opérations inutiles, telles que le traitement des sections vides d'un formulaire. Cette technique améliore la vitesse et l'efficacité, sans avoir besoin de réentraîner entièrement le modèle.

L'**élagage de type 2 sur 4** est **une approche structurée** qui attribue la valeur 0 à 2 paramètres sur 4. Le matériel spécialisé identifie alors rapidement les blocs de paramètres inactifs, puis les contourne efficacement, avec à la clé un temps de calcul réduit et des performances améliorées.



L'évolution des **grands modèles de langage**



À l'origine, les grands modèles de langage (LLM) n'étaient que des expérimentations utilisées à des fins de recherche, qui s'appuyaient essentiellement sur des architectures de transformateurs. Aujourd'hui, ils sont devenus des outils indispensables au fonctionnement d'applications concrètes. Leur taille, qui varie souvent entre des dizaines et des centaines de milliards de paramètres, permet d'atteindre de hauts niveaux de raisonnement, de créativité et de spécificité dans un domaine donné. Ces performances s'obtiennent grâce à un processus appelé inférence.

L'inférence est le processus dans le cadre duquel un modèle entraîné traite de nouvelles données d'entrée pour générer un résultat, comme la prédiction du mot suivant dans une phrase ou l'identification d'un objet dans une image. Pendant l'entraînement, le modèle apprend à partir de grands ensembles de données. Pendant l'inférence, il va exploiter les connaissances acquises pour prendre des décisions en temps réel. Ce processus doit donc être rapide et efficace, en particulier lorsque les modèles sont déployés dans des environnements de production pour faire fonctionner des applications interactives, effectuer des analyses en temps réel ou automatiser des tâches à grande échelle.

Les modèles d'inférence traitent les données d'entrée telles que du texte, des images ou des enregistrements audio sous la forme de jetons textuels, puis les transmettent à des architectures de transformateurs à plusieurs couches qui génèrent des prédictions. Les jetons textuels sont des unités distinctes dans lesquelles les données d'entrée sont décomposées avant d'être traitées par un modèle. Dans les modèles basés sur du texte, les jetons textuels peuvent représenter des caractères individuels, des sous-mots ou des mots entiers, selon la stratégie de conversion en jetons textuels utilisée.

Ces modèles transmettent les jetons textuels d'entrée à des architectures avancées de transformateurs à plusieurs couches qui appliquent une séquence d'opérations mathématiques pour analyser le contexte, mesurer les relations et déterminer les résultats possibles. Chaque couche affine la façon dont le modèle comprend l'entrée pour produire une prédiction, un jeton textuel à la fois. Si cette génération progressive de jetons textuels permet d'obtenir des résultats très précis et adaptés au contexte, elle augmente toutefois l'intensité de calcul des charges de travail d'inférence, en particulier pour les grands modèles comportant de nombreuses couches.

En marge des grands modèles de langage basés sur du texte, il existe des architectures similaires sur lesquelles reposent plusieurs domaine de l'IA, notamment les modèles de vision et les systèmes multimodaux. Les modèles de vision appliquent aux images et aux vidéos des principes de calcul identiques à ceux des transformateurs basés sur des jetons textuels. Au lieu de diviser du texte en jetons textuels, les données en pixels sont converties en incorporations. Ces incorporations capturent des schémas spatiaux, des contours, des textures et des relations entre les éléments visuels qui seront exploités par le modèle pour effectuer des tâches telles que la classification d'images, la détection d'objets, la segmentation et la réponse visuelle à une question. Lorsqu'ils sont déployés en production, les modèles de vision peuvent prendre en charge des cas d'utilisation tels que l'inspection automatisée, l'imagerie médicale et la modération de contenus.

Avec la généralisation de l'utilisation de l'IA au sein des entreprises, les architectures de modèles deviennent toujours plus grandes et complexes. De nouveaux concepts, tels que l'approche MoE (Mixture of Experts), visent à améliorer les performances en activant uniquement certaines parties du modèle pour chaque opération d'inférence, et permettent ainsi de réduire la puissance de calcul requise. Ces innovations ouvrent la voie à des modèles encore plus puissants, tout en aidant à trouver le point d'équilibre entre les performances d'un côté, et les coûts et besoins en énergie de l'autre.

Tous les modèles, indépendamment de leur taille, doivent être mis à disposition et optimisés de manière efficace pour être exploitables en production. C'est pourquoi l'ingénierie des performances de l'inférence devient une priorité pour les entreprises qui cherchent à déployer des modèles.

Les défis liés au **déploiement de l'inférence**

Le déploiement de l'inférence pour de grands modèles pose plusieurs problèmes.

Les modèles contenant des milliards de paramètres nécessitent une mémoire GPU considérable pour stocker leurs pondérations et états intermédiaires, notamment les caches clé-valeur. Avec l'augmentation du nombre de requêtes simultanées ou de la longueur des entrées, les contraintes en matière de mémoire deviennent des goulets d'étranglement critiques qui limitent le débit et la réactivité du modèle. Les méthodes de déploiement de base reposent souvent sur des techniques inefficaces de traitement par lots qui empêchent d'utiliser pleinement les ressources matérielles et augmentent la latence.

De plus, la mise en œuvre de mécanismes d'attention dans les architectures de transformateurs peut solliciter de nombreuses ressources de calcul, en particulier avec des entrées longues, ce qui augmente considérablement les délais de réponse. Ces problèmes peuvent être résolus grâce à des méthodes sophistiquées d'optimisation de l'exécution telles qu'une gestion efficace de la mémoire, des stratégies avancées de traitement par lots et des mécanismes d'attention optimisés comme l'algorithme PagedAttention. En combinant ces méthodes, il est possible d'améliorer les performances et la réactivité des applications en conditions réelles.

Une approche globale pour de **meilleures performances d'inférence**

L'optimisation de l'inférence consiste à améliorer l'efficacité d'exécution d'un modèle d'IA une fois qu'il est déployé en production. L'exécution de grands modèles de langage en production peut rapidement devenir coûteuse, notamment s'il faut gérer de grands volumes de jetons textuels, de longues instructions génératives et des demandes d'utilisation toujours plus nombreuses. **Les coûts liés à l'inférence peuvent être optimisés en réduisant la consommation de mémoire, en augmentant le débit et en diminuant les besoins en matériel, sans pour autant dégrader la précision ou l'expérience utilisateur.**

Alors que l'entraînement des modèles prend généralement la forme d'une tâche à une seule instance (sauf pour le réentraînement d'un modèle), l'inférence s'exécute en continu et génère des résultats en temps réel en réponse aux entrées de l'utilisateur. Dans le cas des LLM et des modèles de vision, l'inférence peut rapidement devenir l'étape la plus coûteuse et la plus gourmande en ressources d'un déploiement de l'IA, en particulier dans une infrastructure hybride ou à échelle mondiale.

L'adoption d'une stratégie d'optimisation globale qui cible à la fois le modèle et l'environnement d'exécution est nécessaire pour déployer efficacement des LLM à grande échelle. Même si l'objectif principal reste d'optimiser les paramètres des modèles à l'aide de la **quantification** et de l'**élagage**, il est possible d'améliorer les performances en affinant le processus de déploiement de l'inférence à l'aide de techniques telles que le **préremplissage par blocs**[4], la **mise en cache des préfixes**[5], le **décodage spéculatif**[6], ainsi que **la désagrégation du préremplissage et du décodage**[7].

4 « **Optimization and Tuning** », *vLLM*, 7 août 2025.

5 « **What is Automatic Prefix Caching?** », *vLLM*, consulté le 8 août 2025.

6 « **How Speculative Decoding Boosts vLLM Performance by up to 2.8x** », *vLLM*, 17 octobre 2024.

7 Kuntai Du, « **vLLM Office Hours - Disaggregated Prefill and KV Cache Storage in vLLM - November 14, 2024** », *YouTube*, 18 novembre 2024.

Présentation des formats de modèle et des environnements d'exécution de l'inférence

Pour distribuer efficacement de grands modèles, il faut choisir un environnement d'exécution adapté à l'inférence car les environnements d'exécution de base sont de véritables goulets d'étranglement. Voici les principaux environnements d'exécution utilisés pour l'inférence :

- **vLLM** : vLLM est une bibliothèque de code Open Source gérée par la communauté vLLM. Ce framework permet aux grands modèles de langage d'effectuer des calculs plus efficaces à grande échelle. Plus précisément, vLLM est un serveur d'inférence qui accélère les résultats des applications d'IA générative en optimisant l'utilisation de la mémoire GPU. Il est très utilisé dans le secteur en raison de son débit élevé et de sa faible latence. Il s'appuie en outre sur des innovations telles que l'algorithme PagedAttention qui permet de traiter efficacement davantage de jetons textuels dans la mémoire GPU.

- **Triton** : souvent considéré à tort comme un environnement d'exécution autonome, Triton fonctionne plutôt comme une API front-end pour divers moteurs back-end, notamment TensorRT et vLLM. Si l'association de Triton et TensorRT permet de bénéficier de performances légèrement supérieures sur les GPU NVIDIA, elle complexifie toutefois la configuration et limite la prise en charge des modèles. Les clients indiquent souvent qu'il est bien plus difficile d'améliorer les performances avec Triton qu'avec vLLM.

- **SGLang** : dérivé de vLLM, le nouveau framework SGLang est optimisé pour des cas d'utilisation spécifiques. Il utilise la plupart des composants sous-jacents de vLLM, mais n'est compatible qu'avec un nombre réduit d'architectures de modèles. Même s'il est plus performant que vLLM dans des contextes précis, sa flexibilité limitée et sa prise en charge communautaire le rendent moins intéressant pour une adoption à l'échelle de l'entreprise.

Une double approche pour des modèles plus efficaces

1. Optimisation de l'exécution de l'inférence (vLLM)

Limites de l'exécution

Comme mentionné dans le chapitre précédent, **il peut être difficile de déployer efficacement de grands modèles de langage en raison des limites inhérentes aux méthodes de base de déploiement de l'inférence**.

Ces limites peuvent inclure une utilisation inefficace de la mémoire GPU, un traitement par lots peu optimal ou une génération ralentie de jetons textuels. Les environnements d'exécution stockent généralement de manière inefficace les données de calcul intermédiaires, telles que des caches clé-valeur, ce qui sollicite beaucoup la mémoire GPU et limite la capacité de traitement des requêtes simultanées. En outre, les stratégies de traitement par lots trop simples laissent des GPU inactifs ou ne les utilisent pas de manière optimale, ce qui réduit considérablement le débit. Par ailleurs, comme les environnements d'exécution de base fonctionnent mal avec les mécanismes d'attention plus lents, la latence augmente lors du traitement de longues séquences d'entrée.

Utilité de vLLM

Le framework vLLM permet de résoudre de nombreux problèmes d'exécution grâce à des techniques avancées spécialement optimisées pour des opérations d'inférence performantes :

- **Traitement par lots continu** : vLLM réduit les temps d'inactivité des GPU en traitant simultanément les jetons textuels issus de différentes requêtes entrantes. Au lieu de traiter une seule requête à la fois, il regroupe les jetons textuels de différentes séquences dans des lots, ce qui améliore considérablement l'utilisation des GPU et le débit d'inférence.

- **PagedAttention** : grâce à ce nouveau mécanisme de gestion de la mémoire, vLLM gère efficacement les caches clé-valeur à grande échelle. Cette technique alloue et gère de manière dynamique les pages de mémoire des GPU, ce qui permet d'augmenter considérablement le nombre de requêtes simultanées traitées et de prendre en charge des séquences beaucoup plus longues sans saturer la mémoire.

Pour en savoir plus, lisez cet **article de blog technique sur vLLM**.

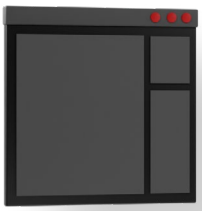
Avantages du déploiement de vLLM

Capacités d'intégration complètes : vLLM permet de charger directement des modèles à partir de référentiels courants tels que Hugging Face et sert de back-end hautes performances au sein de frameworks comme Triton Inference Server. Sa compatibilité avec de nombreuses plateformes matérielles, notamment les GPU NVIDIA, AMD et les TPU Google, simplifie davantage le déploiement à l'échelle de l'entreprise.

Standardisation et indépendance vis-à-vis des fournisseurs : l'utilisation d'un environnement d'exécution courant comme vLLM permet aux entreprises de bénéficier d'une solution standardisée qui garantit des performances fiables dans divers environnements matériels et leur évite toute dépendance vis-à-vis de solutions propriétaires.

Pour mieux comprendre les techniques de parallélisme de vLLM, consultez cet **article de blog technique détaillé**.

2. Optimisation du modèle d'IA



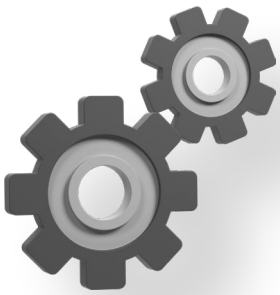
Importance de l'optimisation des grands modèles de langage

L'une des principales difficultés rencontrées en production est la gestion de la mémoire et de l'efficacité du calcul. Les grands modèles sollicitent souvent un volume élevé de mémoire GPU pour stocker les paramètres et le contexte dans le cache clé-valeur, en particulier lorsqu'ils traitent des instructions longues ou plusieurs requêtes simultanées. Si les modèles ne sont pas optimisés, ils risquent de s'exécuter de manière inefficace, ce qui se traduira par une augmentation des coûts d'exploitation. La latence est un autre problème majeur : les utilisateurs attendent des réponses en temps réel, et les retards causés par la taille importante du modèle ou l'inefficacité de l'exécution peuvent dégrader l'expérience ainsi que l'efficacité des workflows en aval.

Utilité de la compression des modèles

La compression d'un modèle permet de résoudre certains des principaux problèmes que rencontrent les entreprises lors du déploiement de l'IA à grande échelle, notamment la rentabilité et l'optimisation des performances.

Comme la taille des modèles augmente et peut atteindre plusieurs milliards de paramètres, leur déploiement nécessite un grand nombre de ressources, notamment en matière de mémoire et de calcul. Les techniques de compression des modèles, telles que la quantification et l'élagage, réduisent légèrement la précision et le nombre de paramètres tout en diminuant considérablement l'empreinte de la mémoire et les besoins en calcul, sans véritablement modifier l'exactitude des résultats. Grâce à la compression des modèles, les entreprises peuvent exécuter des charges de travail d'IA plus efficacement, en limitant le nombre de GPU et d'autres accélérateurs utilisés. Les coûts d'exploitation sont donc considérablement réduits tandis que les opérations d'inférence s'exécutent plus vite, ce qui s'avère indispensable pour les applications qui exigent des réponses en temps réel.



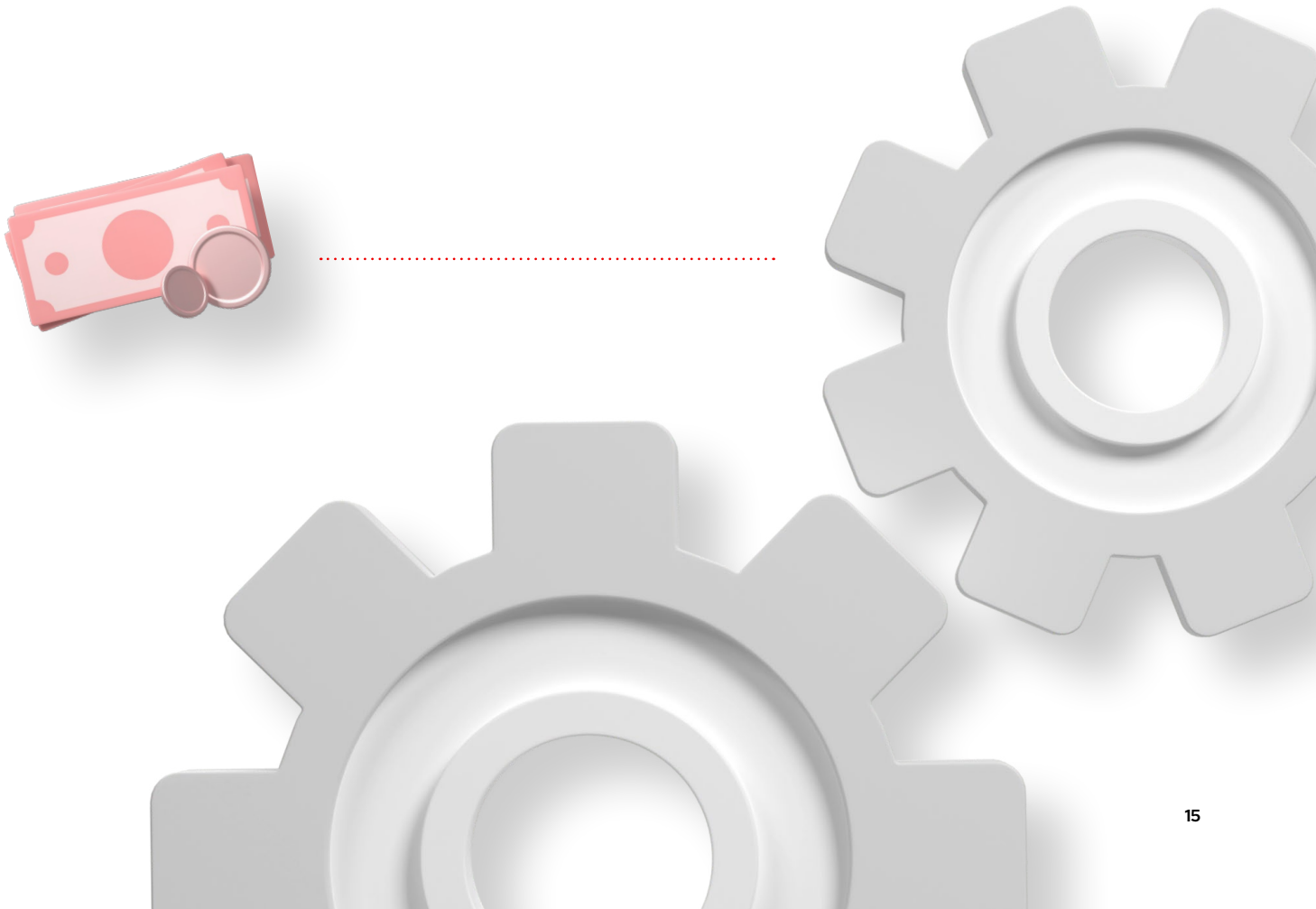
Optimisation des coûts d'un modèle pour l'inférence

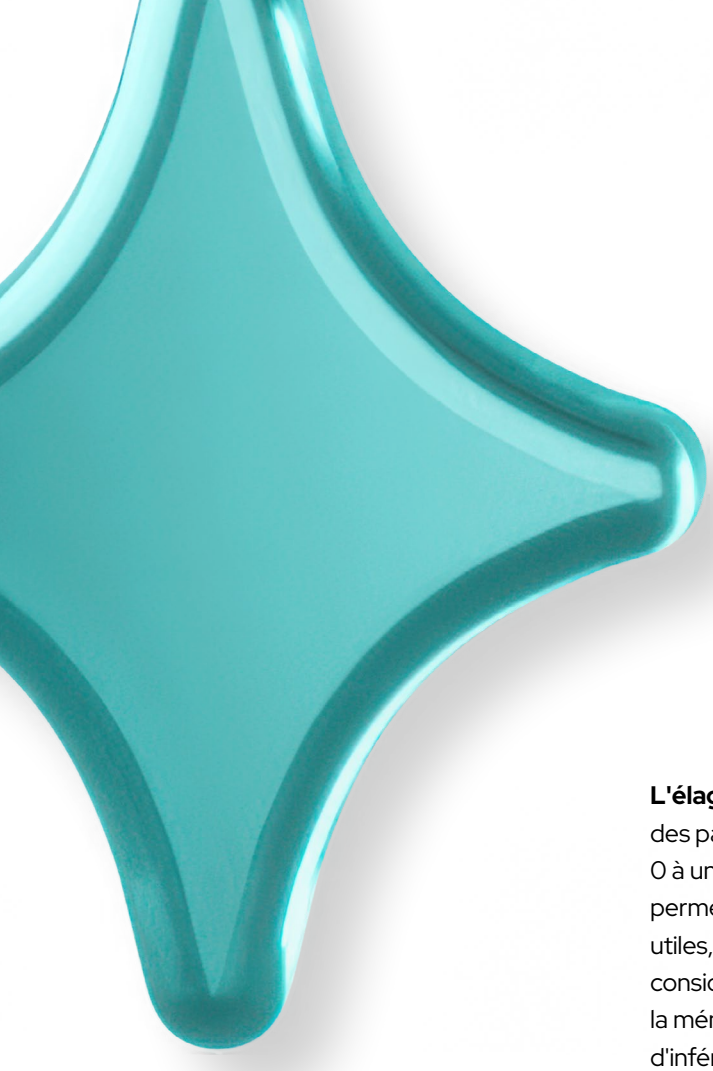
La compression des modèles est l'une des méthodes les plus efficaces pour réduire ces coûts. Les techniques de compression, telles que la quantification et l'élagage, diminuent la taille du modèle ainsi que les besoins en calcul. Elles permettent ainsi d'exécuter les charges de travail d'inférence sur un nombre réduit de GPU ou sur des GPU de plus petite taille.

La **quantification** optimise un modèle en réduisant la précision de ses valeurs numériques, plus précisément ses pondérations et ses activations. En général, les modèles fonctionnent avec une précision de 16 bits (voire 32 bits), dans des formats tels que FP16 ou BF16.

La quantification compresse ces valeurs dans des formats de précision inférieurs, notamment 8 bits (INT8 ou FP8) ou même 4 bits (INT4). Ce processus réduit considérablement la capacité de mémoire nécessaire pour stocker les paramètres du modèle. Ainsi, un modèle tel que Llama qui compte 70 milliards de paramètres et nécessite environ 140 Go de mémoire n'utilisera plus que 40 Go une fois compressé. Cette compression libère non seulement de la mémoire pour des calculs supplémentaires, mais elle améliore également le débit, en particulier dans les situations qui dépendent de la capacité de mémoire disponible. Un GPU avec 48 Go de VRAM pourra par exemple gérer un modèle de 40 Go plus rapidement qu'un modèle de 140 Go.

Toutefois, la précision peut être dégradée si la quantification est trop intensive en raison de la perte d'informations. Pour remédier à ce problème, la quantification fine utilise des facteurs de mise à l'échelle qui préservent la précision du modèle, avec à la clé une dégradation souvent inférieure à 1 %. La quantification permet de doubler la vitesse de calcul en optimisant l'utilisation du matériel, et ainsi de réduire significativement la latence et les coûts d'exploitation.





L'élagage optimise un modèle par le biais de la réduction structurée des paramètres, qui repose essentiellement sur l'attribution de la valeur 0 à une grande partie des pondérations du modèle. Cette technique permet d'identifier et d'éliminer les pondérations redondantes ou moins utiles, et ainsi de simplifier les calculs pendant l'inférence. L'élagage peut considérablement simplifier le modèle, et donc diminuer l'utilisation de la mémoire ainsi que la charge de calcul, avec à la clé des opérations d'inférence plus rapides et des coûts d'exploitation réduits.

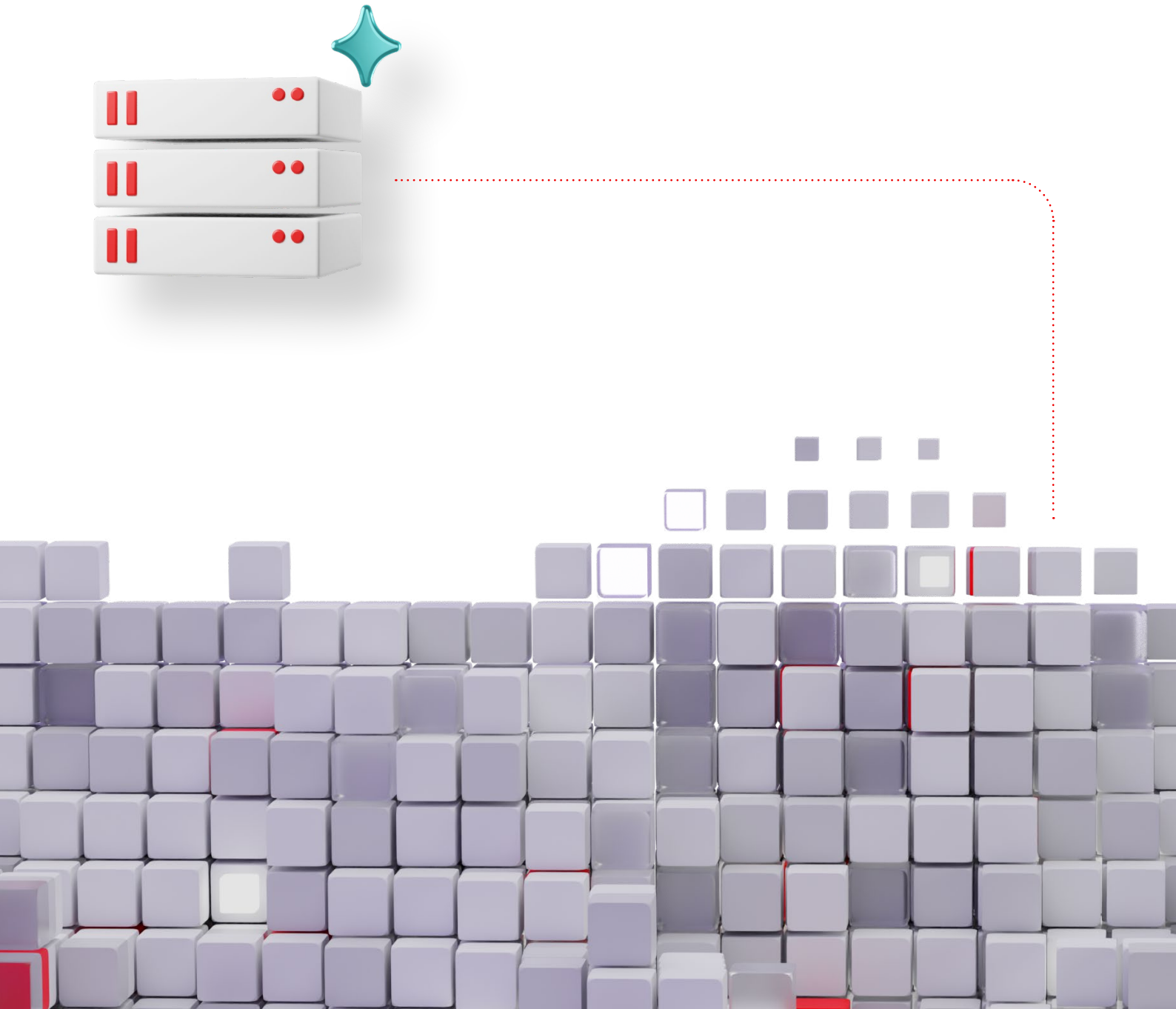
Pour un élagage efficace, il faut néanmoins réentraîner le modèle, une étape de calcul intensive qui mobilise de nombreuses ressources en amont. L'efficacité de l'élagage dépend des capacités matérielles. Par exemple, l'élagage semi-structuré est pris en charge par les accélérateurs modernes tels que les GPU, qui utilisent des modèles spécifiques de pondérations nulles pour accélérer les calculs. Son principal avantage réside dans sa capacité à réduire considérablement les besoins en calcul, à partir du moment où il est correctement mis en œuvre.

Si l'élagage peut offrir des avantages considérables, en particulier lorsqu'il est associé à d'autres méthodes d'optimisation comme la quantification, il nécessite généralement un processus plus complexe. C'est pourquoi cette technique est plutôt recommandée pour les situations impliquant une mise à l'échelle importante ou des configurations matérielles spécialisées. Appliqué de manière rigoureuse, l'élagage permet aux entreprises d'améliorer l'efficacité de l'inférence. Mais parce que cette technique est assez complexe, c'est plus généralement la quantification qui est recommandée.

Grâce à l'adoption de workflows de compression et d'environnements d'exécution validés, les entreprises peuvent mieux gérer les coûts d'exploitation, favoriser l'évolutivité et se préparer à l'augmentation future de l'utilisation de l'IA sans mobiliser trop de ressources d'infrastructure.

Préservation du niveau de précision

Si les techniques de compression de modèles telles que la quantification et l'élagage réduisent les besoins en matière de mémoire et de calcul, elles sont aussi spécifiquement conçues pour maintenir des niveaux de précision acceptables. La quantification sur 8 bits offre par exemple une précision proche de celle de référence, tout en utilisant deux fois moins de mémoire. Les modèles sur 4 bits peuvent également préserver un haut niveau de performances s'ils sont optimisés à l'aide de techniques de quantification avancées telles que l'arrondi et l'étalonnage des pondérations. Les modèles d'élagage structuré, tels que l'élagage de type 2 sur 4, permettent aux accélérateurs matériels d'éviter les opérations redondantes sans dégrader la qualité des résultats. Dans de nombreux scénarios de production, les équipes parviennent à diminuer significativement le nombre de ressources nécessaires avec une réduction minimale voire nulle des performances du modèle. Les tests et la validation restent des étapes essentielles, mais pour la plupart des applications, la bonne mise en œuvre de la compression permet d'obtenir des performances d'inférence élevées avec une précision inchangée.

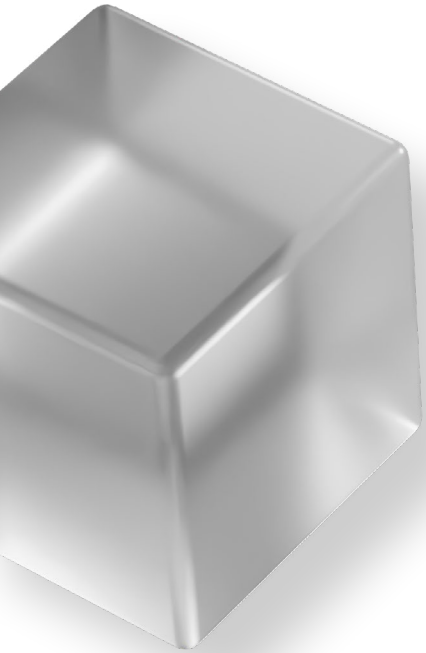


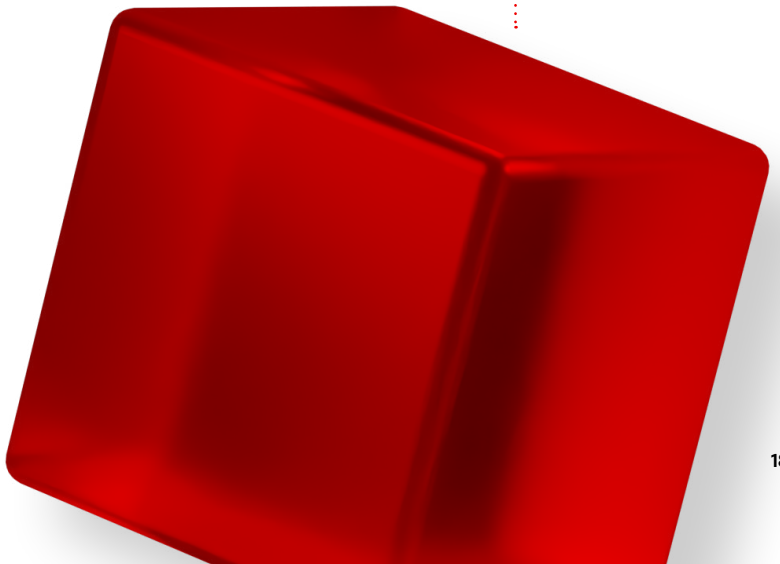
Red Hat AI

Présentation de Red Hat AI

Red Hat AI est une plateforme qui accélère l'innovation en matière d'IA et qui réduit les coûts d'exploitation liés au développement et à la distribution de solutions d'IA dans les environnements de cloud hybride. Cette plateforme évolutive et flexible peut simplifier l'intégration des données privées, réduire les coûts grâce à des modèles optimisés et des opérations d'inférence efficaces, ainsi qu'accélérer la distribution de workflows d'IA agentique.

Red Hat AI permet aux entreprises de gérer et surveiller le cycle de vie des modèles d'IA prédictive et générative à grande échelle, des déploiements sur un seul serveur aux plateformes distribuées à très grande échelle. La plateforme s'appuie sur des technologies Open Source et les solutions d'un écosystème de partenaires qui sont axées sur les performances, la stabilité et la prise en charge des GPU au sein d'infrastructures variées.





L'offre Red Hat AI inclut les éléments suivants :

- **Des modèles optimisés et validés** : ces modèles préévalués aux performances vérifiées facilitent les tests et le réglage fin.
- **LLM Compressor** : cette boîte à outils facilite l'application de la quantification et de la compression aux modèles courants, réduisant ainsi les ressources nécessaires pour l'inférence sans perte de précision.
- **Des fonctions de personnalisation des modèles** : ces outils permettent d'ajuster ou adapter les modèles de fondation aux besoins spécifiques de l'entreprise.
- **Un environnement d'exécution de l'inférence hautes performances** : cet environnement d'exécution basé sur vLLM et optimisé utilise des techniques avancées de traitement par lots et de gestion de la mémoire pour une distribution des modèles plus fiable, évolutive et efficace.
- **Des outils LLMOps** : ces pratiques et outils rationalisent le déploiement, la surveillance et la gestion des LLM dans les environnements de production.
- **Des outils de sécurité et d'évaluation de l'IA** : ces frameworks et méthodes permettent d'évaluer la précision, l'équité et la robustesse des modèles afin de garantir la responsabilité et la fiabilité des déploiements de l'IA.
- **Une mise à l'échelle flexible et cohérente** : la prise en charge de l'infrastructure garantit la flexibilité et la cohérence lors de la mise à l'échelle de l'IA dans des environnements de cloud hybride.
- **Des fonctionnalités de distribution accélérée de l'IA agentique** : ces fonctionnalités sont conçues pour déployer rapidement des systèmes d'IA avancés et autonomes, permettant ainsi aux entreprises de rester à la pointe de l'innovation en matière d'IA.

Optimisation des modèles avec Red Hat AI

Red Hat AI permet aux entreprises d'optimiser leurs modèles d'IA à l'aide de techniques avancées conçues pour trouver le point d'équilibre entre efficacité, précision et rentabilité.

Red Hat AI met l'accent sur deux grands aspects de l'optimisation des modèles : l'efficacité de l'exécution et la compression des modèles. En associant ces approches, la gamme de produits Red Hat pour l'IA permet d'obtenir des opérations d'inférence rapides et efficaces tout en réduisant le nombre de ressources de calcul nécessaires. Plus précisément, la solution Red Hat AI Inference Server utilise des méthodes de traitement par lots continu ainsi que de gestion efficace de la mémoire. Les modèles peuvent traiter plus de jetons textuels par seconde, ce qui permet d'atteindre un débit plus élevé en sollicitant moins les GPU.

L'outil Red Hat AI LLM Compressor offre une approche standardisée pour l'application des techniques de compression décrites dans ce livre numérique, et vise à fournir une optimisation qui préserve la précision à 99 %. Il permet de générer des versions optimisées de modèles courants qui sont réglés pour des environnements d'exécution de l'inférence tels que vLLM. Il facilite ainsi l'exécution de modèles compressés hautes performances sur un grand nombre de configurations matérielles.

Red Hat AI fournit des capacités de validation complètes pour aider les entreprises à sélectionner, déployer et mettre à l'échelle des modèles optimisés en toute confiance. Compte tenu de la multitude de LLM disponibles, les entreprises ont souvent du mal à identifier les modèles qui correspondent le mieux à leurs cas d'utilisation en matière de précision, de performances et de rentabilité. Pour relever ces défis, Red Hat AI utilise des outils de validation Open Source (tels que GuideLLM, Language Model Evaluation Harness et vLLM) pour évaluer rigoureusement les performances des modèles dans le cadre de différentes tâches d'évaluation. Cette validation prend en compte la reproductibilité et une sélection fiable de modèles pour réduire la complexité et l'incertitude.

Red Hat AI propose également des **recommandations concernant les capacités** afin d'aider les entreprises à préparer avec exactitude leur infrastructure d'IA et à optimiser l'utilisation des ressources. Elles sont alors en mesure de résoudre les problèmes courants, notamment la sous-utilisation du matériel, les coûts de calcul élevés et les inefficacités lors de l'inférence. Cette combinaison de modèles validés, de paramètres de déploiement optimisés et de recommandations matérielles sur mesure permet aux entreprises de renforcer la flexibilité, d'accélérer les déploiements et d'obtenir des performances prévisibles, tout en gérant efficacement les coûts.

Grâce à des techniques de compression et à des environnements d'exécution optimisés, Red Hat AI facilite le déploiement de grands modèles de langage à grande échelle. Les équipes peuvent ainsi répondre à l'augmentation de la demande tout en maîtrisant les coûts, la complexité et l'utilisation des ressources de calcul.

Étapes suivantes

Vous souhaitez réduire le coût et la complexité du déploiement des grands modèles de langage ? Apprenez-en plus sur **Red Hat AI Inference Server** ou contactez votre représentant Red Hat.