Résumé de l'article

Gestion Efficace de la Mémoire pour le Service de Modèles Linguistiques de Grande Taille avec PagedAttention

Lontsi Lambou R

16-12-2024

I. Introduction

Les modèles de langage de grande taille (LLM) ont révolutionné le traitement automatique des langues et d'autres domaines de l'intelligence artificielle. Cependant, leur déploiement et leur utilisation posent de nombreux défis, notamment :

- Une consommation mémoire exponentielle avec la taille du modèle.
- Une infrastructure coûteuse pour l'inférence et la formation.
- La difficulté d'exécuter ces modèles sur des appareils aux capacités limitées.

Cet article propose **PagedAttention**, une méthode de gestion mémoire optimisée pour le déploiement des LLM, en introduisant un mécanisme de *pagination* pour gérer efficacement les contextes d'attention.

II-Contexte

dans la communaute scientifique l'implementation des grands modeles de langages (LLMs) sont influence par l'acces en memoires cache des cles valeurs pour chaque requête sont tres énorme et croît et décroît dynamiquement. et se rétrécit dynamiquement.dans l'architecture de base des (LLMs)

- Encodeur-Décodeur
- Attention Mécanisme
- Couches
  Chaque couche du modèle comprend :
  - Un mécanisme d'attention multi-tête.
  - Un réseau de neurones feedforward positionnel.
  - Des normalisations de couche (layer normalization).
  - Des connexions résiduelles (residual connections) pour faciliter l'apprentissage.
- Embedding

dans l'architecture le mecamisme d'attention utilise la menoire cache pour permettre au modele de se sourvenir des contexte importante a traveers les couches

Question scientifique:
comment mettre la cache cle-valeur (KV-cache) dispomible pour la mise en lot d'un nombre suffisant de demande a la fois dans les grands modeles de langage (LLMs)?

Motivation Eviter le gaspillage de la memoire cache cle valeur

Hypothèse: PagedAttention permettra une meilleure gestion de la mémoire KV, conduisant à une augmentation significative du débit de service des LLM.

III-Revue de littérature

- FasterTransformer:
  - Présentation :
    - Une bibliothèque d'optimisation pour les modèles de type Transformer.
    - Conçue pour améliorer les performances des LLMs en exploitant des noyaux GPU personnalisés.
    - Phase autorégressive : Chaque token dépend des précédents, ralentissant les calculs.
  - Avantage
    - Prend en charge des techniques comme la fusion des opérations matricielles pour réduire la latence.
    - Offre un support pour le batching dynamique afin de traiter simultanément des requêtes de longueurs différentes.
  - limitation:
    - La gestion des caches KV est statique, entraînant une sous-utilisation de la mémoire GPU.
    - Les problèmes de fragmentation mémoire interne et externe ne sont pas complètement résolus.
    - Performances limitées pour des séquences longues et des algorithmes de décodage complexes comme beam search.

III-Revue de littérature

- Orca:
  - Présentation :
    - est une approche d'optimisation de l'attention dans les modèles Transformer, conçue pour réduire les coûts de calcul et de mémoire tout en préservant l'efficacité sur des séquences longues.
    - Plutôt que de stocker toutes les clés/valeurs, ORCA utilise des projections ou des agrégations pondérées pour représenter les informations essentielles.
  - Avantage
    - Implémente un partage des caches KV pour réduire la consommation mémoire.
    - ORCA applique une compression adaptative aux clés (K) et valeurs (V) pour limiter leur croissance mémoire lorsque la séquence s'allonge.
  - limitation:
    - Implémente un partage des caches KV pour réduire la consommation mémoire.
    - Les performances déclinent significativement pour des séquences longues ou pour des modèles volumineux.
    - Le support des algorithmes complexes comme beam search est limité.

III. Problème du KV cache et mémoire inefficace

- **Travaux précédents:** Les systèmes existants souffrent de fragmentation et de duplication redondante de la mémoire KV, limitant la taille des lots.
- **Lacunes dans la recherche:** Gestion inefficace de la mémoire KV dynamique. Manque de partage de la mémoire KV entre les requêtes.
- **Positionnement:** Cette recherche propose une solution nouvelle inspirée des techniques de gestion de mémoire des systèmes d'exploitation.

IV. Solution : PagedAttention

- un nouveau algorithme d'attention, Page- dAttention, et nous construisons un modele LLM, vLLM, pour relever les défis décrits plus haut
- **Fonctionnement** :
  Contrairement aux algorithmes d'attention traditionnels, PagedAttention permet de stocker des clés et des valeurs continues dans des zones non contiguës.
  En particulier, PagedAttention partitionne le cache KV de chaque séquence en blocs KV. Chaque bloc contient les vecteurs de clés et de valeurs pour un nombre fixe de jetons1 , que nous désignons par KV
  - Bloc logique : Tokens logiques dans une séquence.
  - Bloc physique : Mémoire réelle utilisée sur GPU.
  - Correspondance assurée par une table de mappage dynamique.

$$K_j = (k_{(j-1)B+1}, \ldots, k_{jB}), \quad V_j = (v_{(j-1)B+1}, \ldots, v_{jB})$$

IV.Solution: PagedAttention

$$A_{ij} = \left(a_{i,(j-1)B+1}, \ldots, a_{i,jB}\right), \tag{1}$$

$$o_i = \sum_{j=1}^{\lceil i/B \rceil} V_j A_{ij}^T, \tag{2}$$

$$A_{ij} = \frac{\exp(q_i^T K_j/\sqrt{d})}{\sum_{t=1}^{\lceil i/B \rceil} \exp(q_i^T K_t/\sqrt{d})}. \tag{3}$$

où :
- A_{ij} est un vecteur ligne des scores d'attention pour le j-ième bloc KV. - q_i représente la requête pour la position i. - K_j et V_j sont respectivement les clés et les valeurs associées au j-ième bloc. - B est la taille du bloc. - d est la dimension de l'espace latent (ou la taille des vecteurs d'encodage). l'algorithme PagedAttention permet aux blocs KV d'être stockés une mémoire physique non contiguë, ce qui permet une gestion plus souple de la mémoire paginée dans vLLM.

V. Applications et algorithmes de décodage

- **Parallel Sampling** : Partage des blocs KV des prompts initiaux.
- **Beam Search** : Blocs partagés dynamiquement entre candidats.

VI. Kernel Microbenchmark

- Objectif : Mesurer les performances de PagedAttention en isolation pour évaluer le coût des noyaux GPU.
- Résultats :
  - PagedAttention réduit les frais des lancements de noyaux GPU grâce à la fusion des opérations.
  - Meilleure utilisation des blocs mémoire, avec un débit amélioré par rapport aux solutions traditionnelles.

VII. Performance sur des cas réels (Chatbot et Traduction)

- **Chatbot** :
  - Configuration avec des séquences longues et multiples requêtes simultanées.
  - Résultats : Débit multiplié par 2x à 4x par rapport à FasterTransformer.
- **Traduction automatique** :
  - Séquences variables et complexité accrue dans les calculs d'attention.
  - Observation : Gestion optimale des blocs grâce à PagedAttention, sans pénalités de latence.

VIII. Impact de la taille des blocs (suite)

- Analyse approfondie des tailles de blocs fixes vs dynamiques.
- **Constat** :
  - Des blocs plus petits améliorent l'efficacité mémoire mais augmentent légèrement la latence.
  - Des blocs plus grands minimisent les lancements GPU mais augmentent l'utilisation mémoire.
- **Optimisation** : Choix dynamique de la taille des blocs en fonction du workload.

IX. Comparaison avec Orca

- **Orca** : Alternative pour l'optimisation de l'attention.
- **Comparaison des performances** :
  - PagedAttention offre des gains de débit supérieurs (jusqu'à 2x) grâce à une meilleure répartition des blocs.
  - Utilisation mémoire : PagedAttention permet de traiter jusqu'à 22x plus de requêtes avec une mémoire GPU fixe.

Comparaison avec les modeles precedent

Caractéristique	PagedAttention (vLLM)	FasterTransformer	Orca
Gestion des caches KV	Blocs paginés et dynamiques	Allocation statique	Partage basique
Fragmentation mémoire	Minimisée	Présente	Présente
Flexibilité des séquences	Très élevée	Moyenne	Moyenne
Algorithmes de décodage	Sampling parallèle, Beam search optimisé	Basique	Basique
Gains de performance	2-4x (vs autres systèmes)	Modéré	Modéré

X. Optimisations de la mémoire et applications à d'autres workloads

- **Techniques d'optimisation** :
  - Réduction des copies mémoire.
  - Compaction des blocs non utilisés pour minimiser la fragmentation.
- **Applications à d'autres workloads** :
  - Serveurs web : Réduction de la latence des réponses pour des requêtes simultanées.
  - Workloads machine learning : Optimisation des calculs pour des architectures multi-GPU.
  - Stockage distribué : Gestion efficace des métadonnées avec PagedAttention.

XI. Optimisations GPU spécifiques

- Fusion des opérations pour minimiser les frais des lancements GPU.
- Lecture/écriture par blocs pour éviter les accès non contigus à la mémoire.

XII. Évaluation et résultats

- PagedAttention dépasse FasterTransformer et Orca :
  - Débit multiplié par 2x à 4x.
  - Gestion efficace des requêtes sans pertes de latence.
  - Exemple : Jusqu'à 22x plus de requêtes simultanées possibles.

XIII. Conclusion

PagedAttention améliore considérablement la gestion de la mémoire et le débit des LLMs. Cette approche est particulièrement efficace pour les modèles volumineux, les séquences longues et des algorithmes complexes comme le beam search.

Références

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). *Attention is All You Need*.
2. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., Stoica, I. (2023). *Efficient Memory Management for Large Language Model Serving with PagedAttention*. Publié le 12 septembre 2023.
3. Yvon, F. (2020). *Le modèle Transformer: un "couteau suisse" pour le traitement automatique des langues*. Traitement Automatique des Langues, 61(2), 21–54.