StarONNX : Un ordonanceur dynamique pour une inférence rapide et à haut débit sur des ressources hétérogènes

Olivier Beaumont, Jean-François David, Lionel Eyraud-Dubois, Samuel Thibault

StarONNX :

un ordonnanceur dynamique pour une inférence rapide et à haut débit sur des ressources hétérogènes

Olivier Beaumont, Jean-François David, Lionel Eyraud-Dubois, Samuel Thibault

Inria Centre de l'Université de Bordeaux
Université de Bordeaux

03/07/2024

Introduction

- L'inférence des modèles de réseaux de neurones profonds (DNN) sur des processeurs hétérogènes est un défi
- Objectifs : Minimiser la latence et maximiser le débit
- StarONNX : Solution intégrant ONNX Runtime avec StarPU pour l'ordonancement des tâches d'inférence
- Capacité d'utiliser un DNN divisé en deux parties (ou plus) afin de permettre à la fois l'utilisation du GPU et du CPU

Motivations

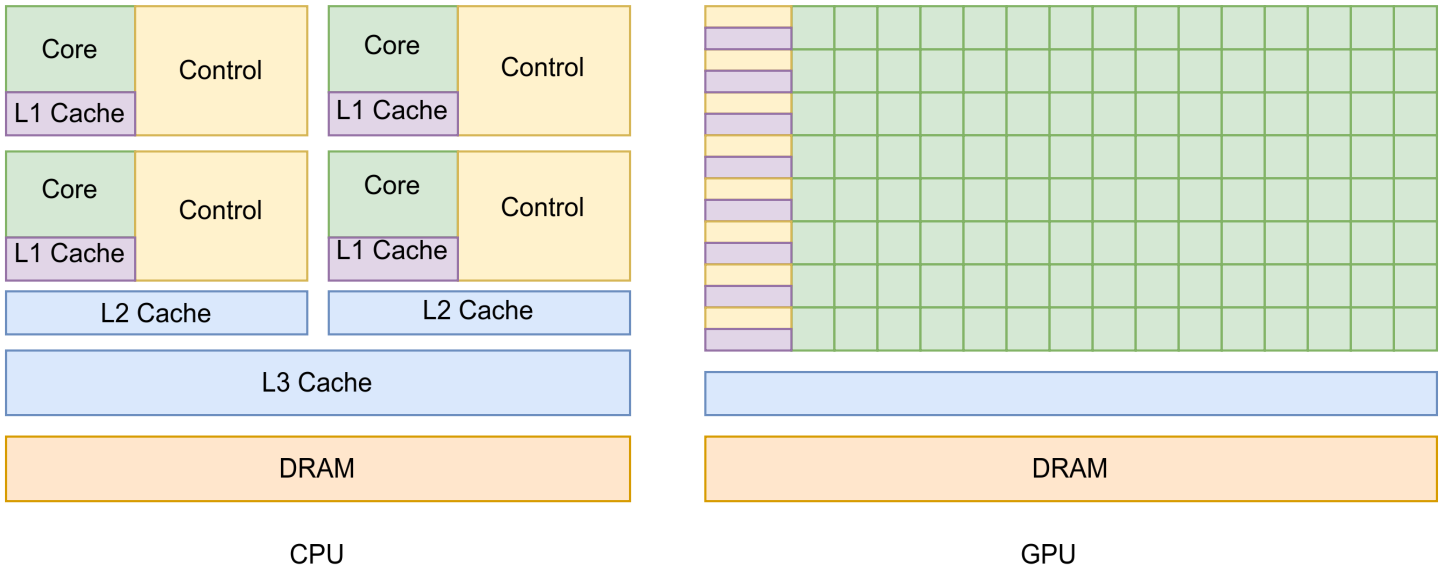
- **Optimisation de l'inférence :**
  - Approches actuelles reposent sur une meilleure utilisation des GPU
  - D'autres solutions utilisent la structure parfois parallélisable des DNN pour permettre l'intervention du CPU
- **Limites des approches actuelles:**
  - Dépendance excessive aux GPU entraîne plusieurs défis
  - Saturation rapide des GPU
  - Sous-utilisation des CPU
- **Problématiques identifiées:**
  - Explosion du débit et impact sur la latence
  - Thompson et al. montrent une augmentation des exigences en matière de calcul, rendant les approches actuelles prohibitives techniquement, économiquement et environnementalement

ONNX Runtime, StarPU et StarONNX

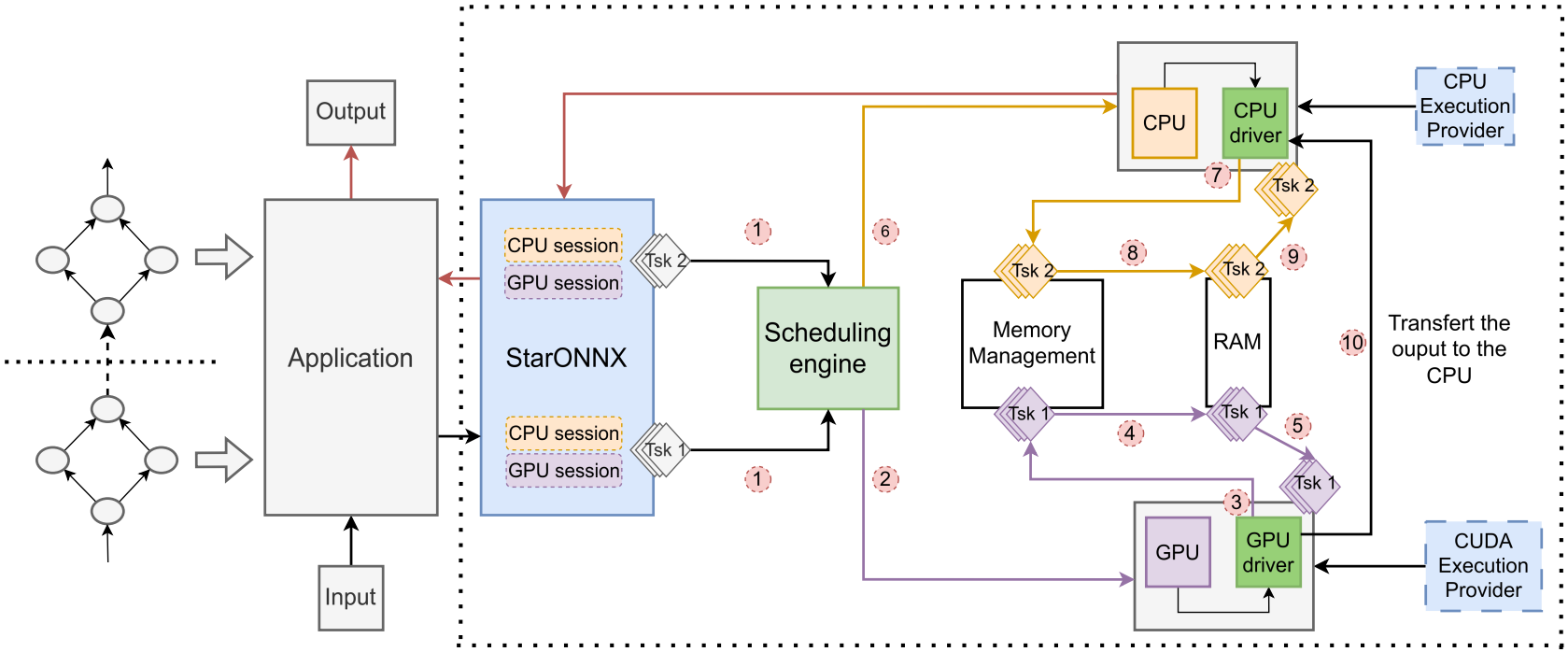
- **ONNX Runtime**
  - Support de plusieurs architectures de processeur
  - Support plusieurs framework DNN
  - Optimisations des modèles DNN
- **StarPU**
  - Exécution pour CPU, GPU, et autres
  - Ordonnancement des tâches
  - Gestion de la mémoire et des transferts de données
- **StarONNX**
  - Intégration d'ONNX Runtime avec StarPU
  - Ordonnancement des tâches d'inférence et gestion des ressources de calcul
  - Exécution et partage dynamique de modèle DNN entre GPU et CPU

Les défis de l'hétérogénéité

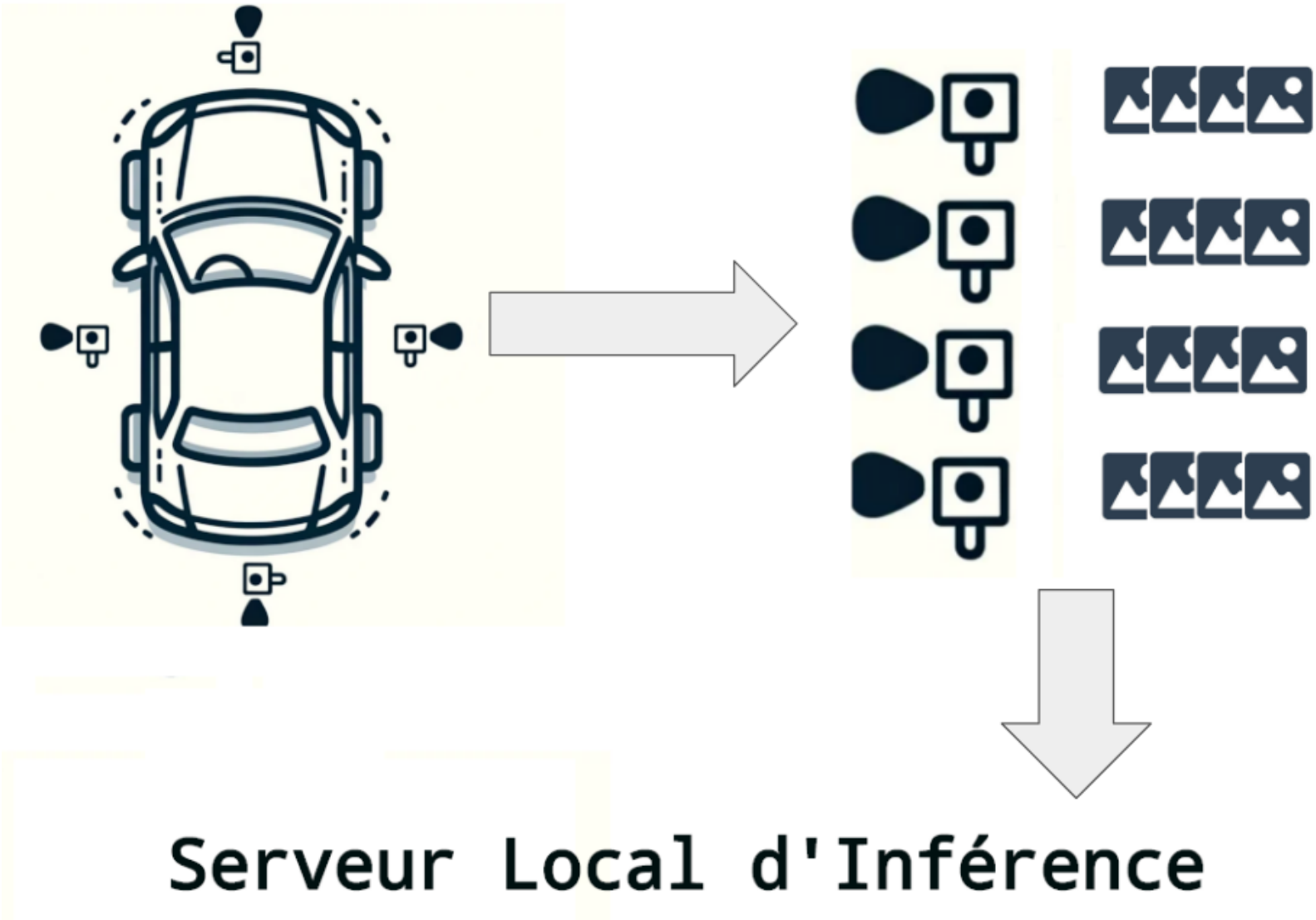
- Problèmes d'architecture et de communication
- Utilisation équilibrée et adaptée des ressources
- Ordonancement des tâches
- Interopérabilité des logiciels



Intégration d'ONNX Runtime dans StarPU



Scénario

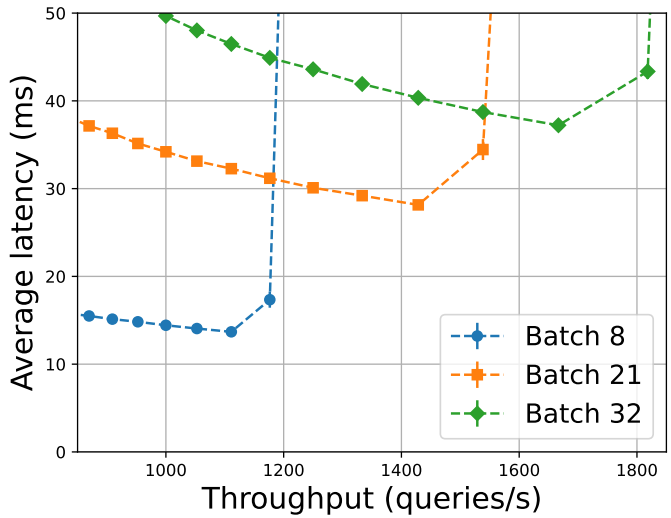


Regroupement des requêtes d'inférences en lots

- Les requêtes d'inférence arrivant au serveur sont regroupées en lots
- Le temps nécessaire pour constituer un lot de N requêtes est

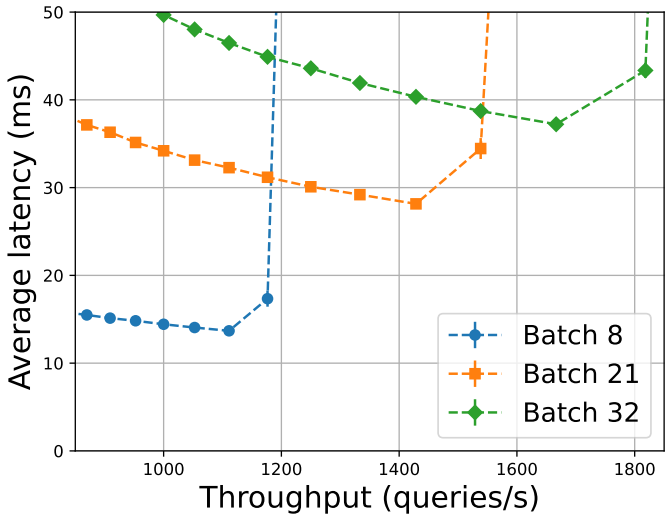
$$T_{\text{lot}} = T \times (N - 1)$$

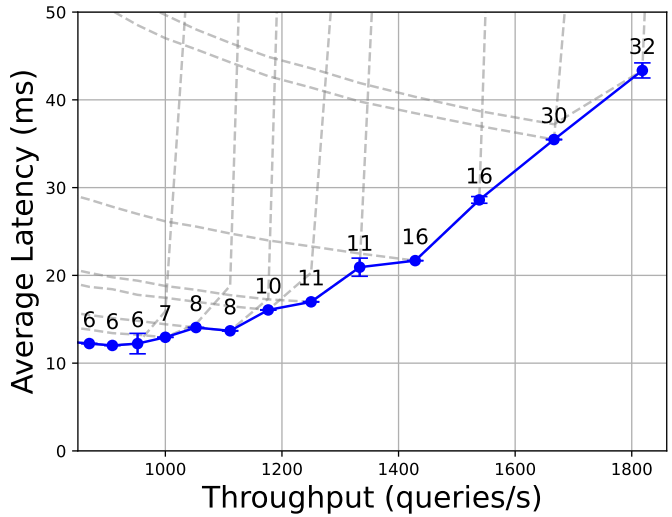
- La latence correspond au temps de constitution d'un lot ajouté aux temps de transfert et de calcul



Évaluation des Performances

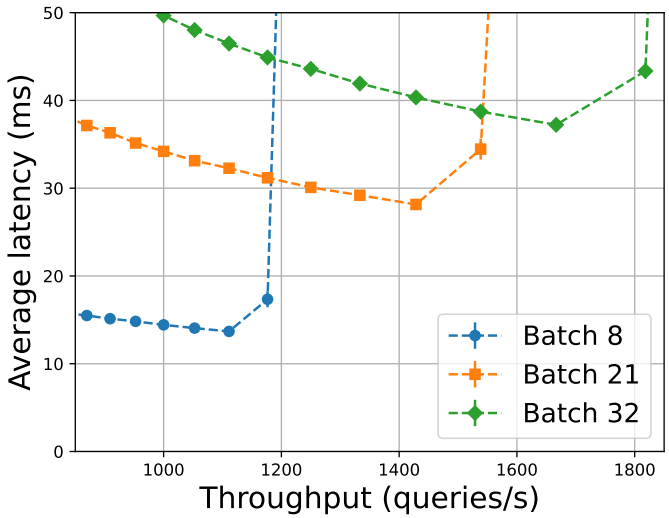
- Mesure de la latence et du débit pour différentes tailles de lot
- Sélection du lot minimisant la latence à un débit fixé

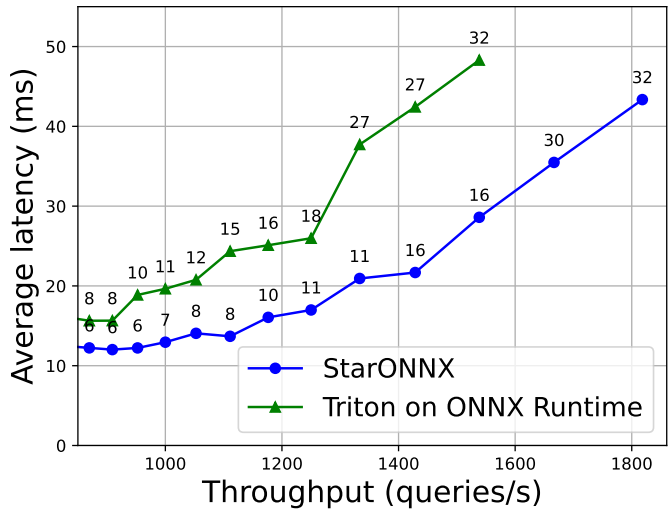




Évaluation des Performances

- Mesure de la latence et du débit pour différentes tailles de lot
- Sélection du lot minimisant la latence à un débit fixé
- Comparaison des performances entre StarONNX et Triton Inference Server





Évaluation des Performances

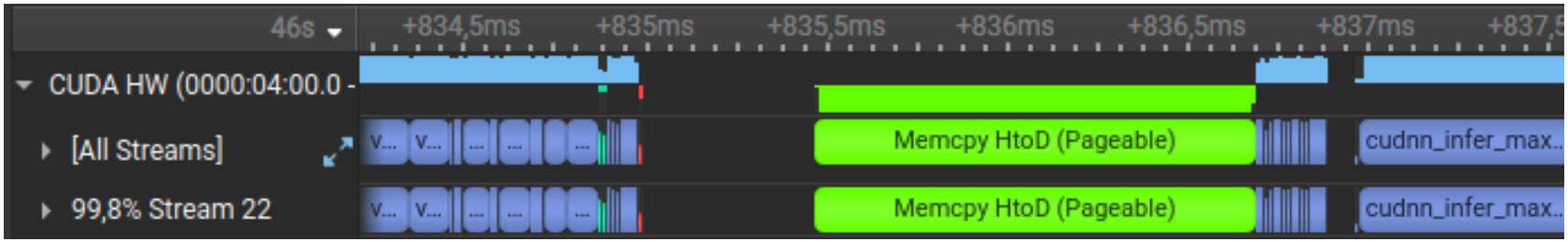


Figure 1: Triton Inference Server

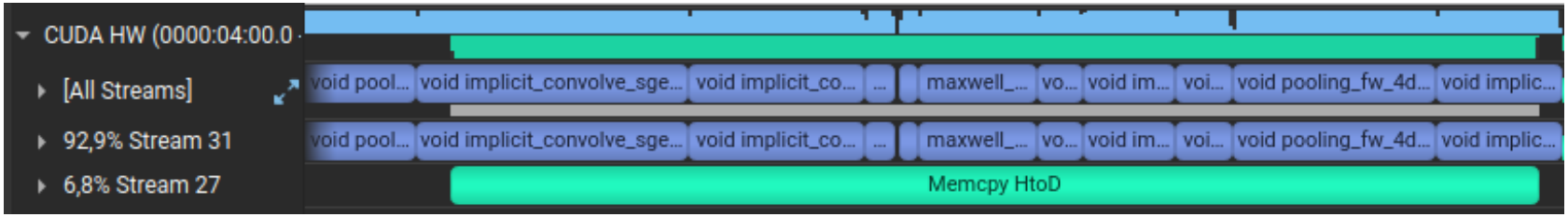


Figure 2: StarONNX

Partitionnement des Modèles DNN

- **Objectif**
  - Utiliser à la fois le GPU et le CPU
  - Suivre le débit fixé et minimiser la latence
- **Avantages**
  - **Optimisation de l'utilisation des Ressources**
  - **Adaptation des Charges de calcul** : GPU pour calculs lourds, CPU pour les plus légers
  - **Ordonnancement dynamique** avec StarPU
  - **Réduction de la Latence** et amélioration du débit

Résultats

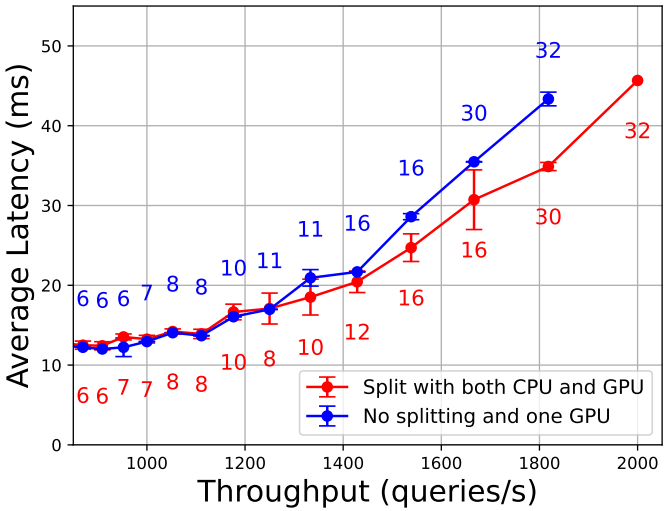


Figure 3: GoogLeNet

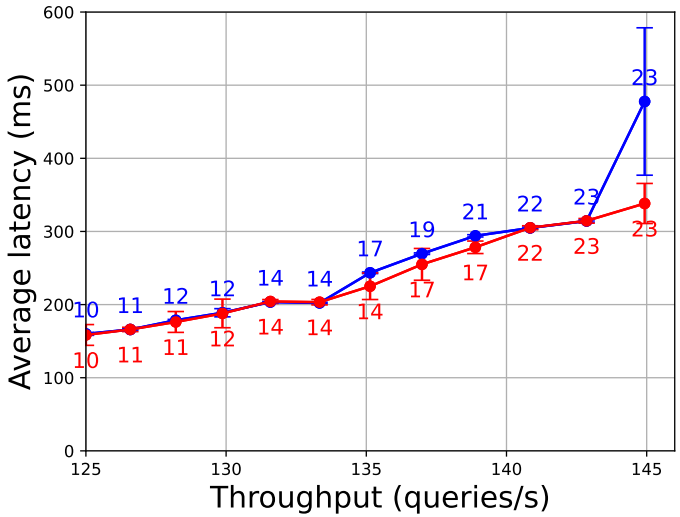


Figure 4: EfficientNet V2

Résultats

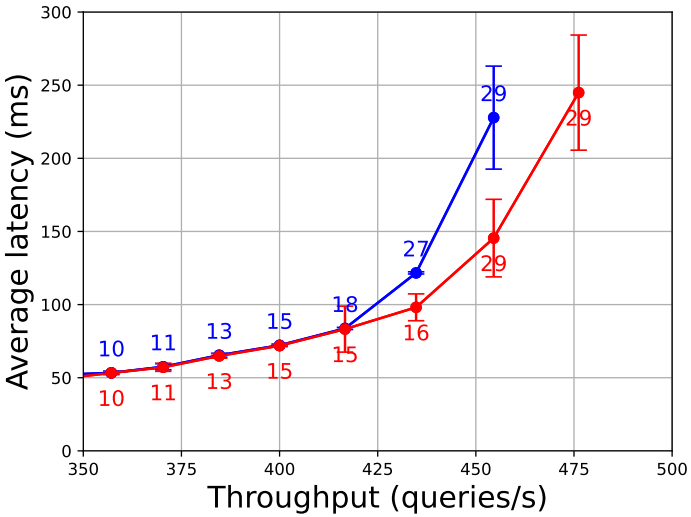


Figure 5: NFNet

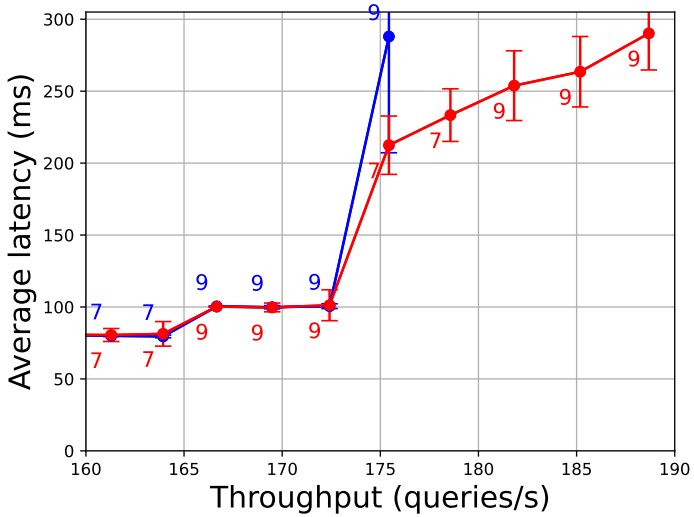


Figure 6: ViT-Face-Expression

Conclusion

- StarONNX : Ordonnanceur dynamique pour les tâches d'inférence sur des ressources hétérogènes.
- Amélioration de l'utilisation des ressources et réduction de la latence par rapport à Triton Inference Server.