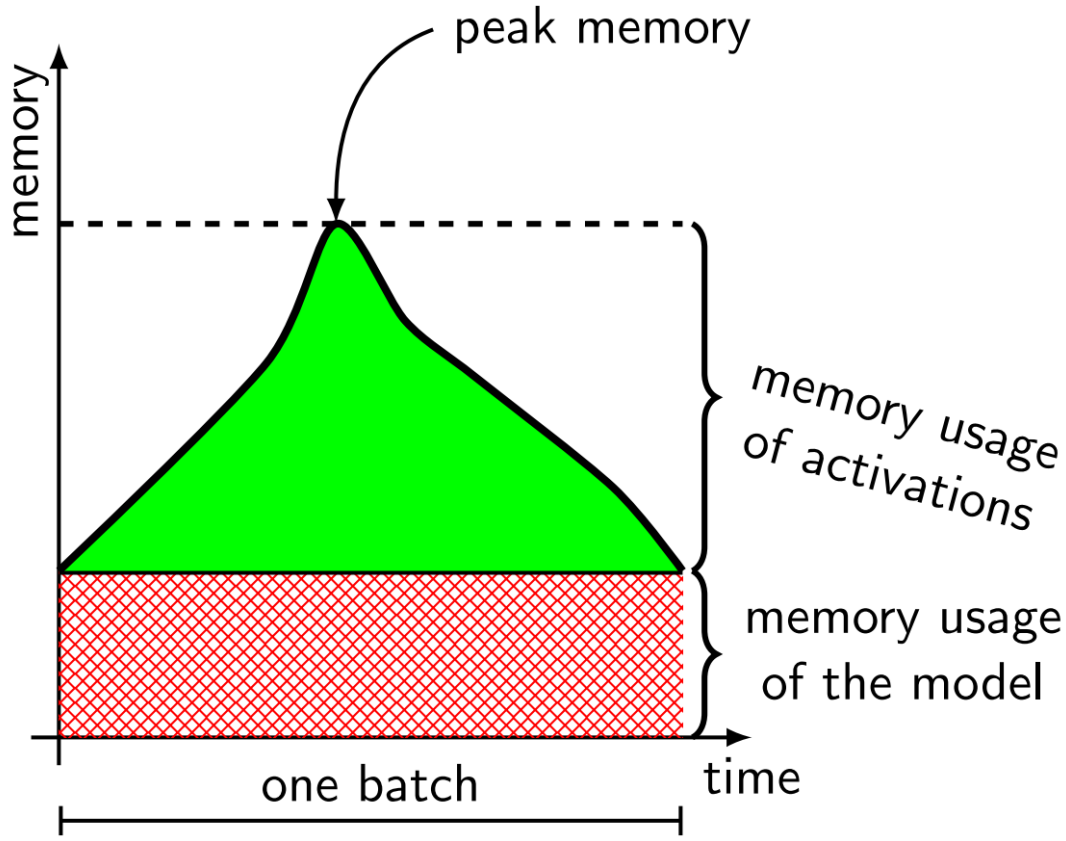
ROTOR-Pipe

Encadrants : Olivier Beaumont, Lionel Eyraud-Dubois et Pierre Lemarinier

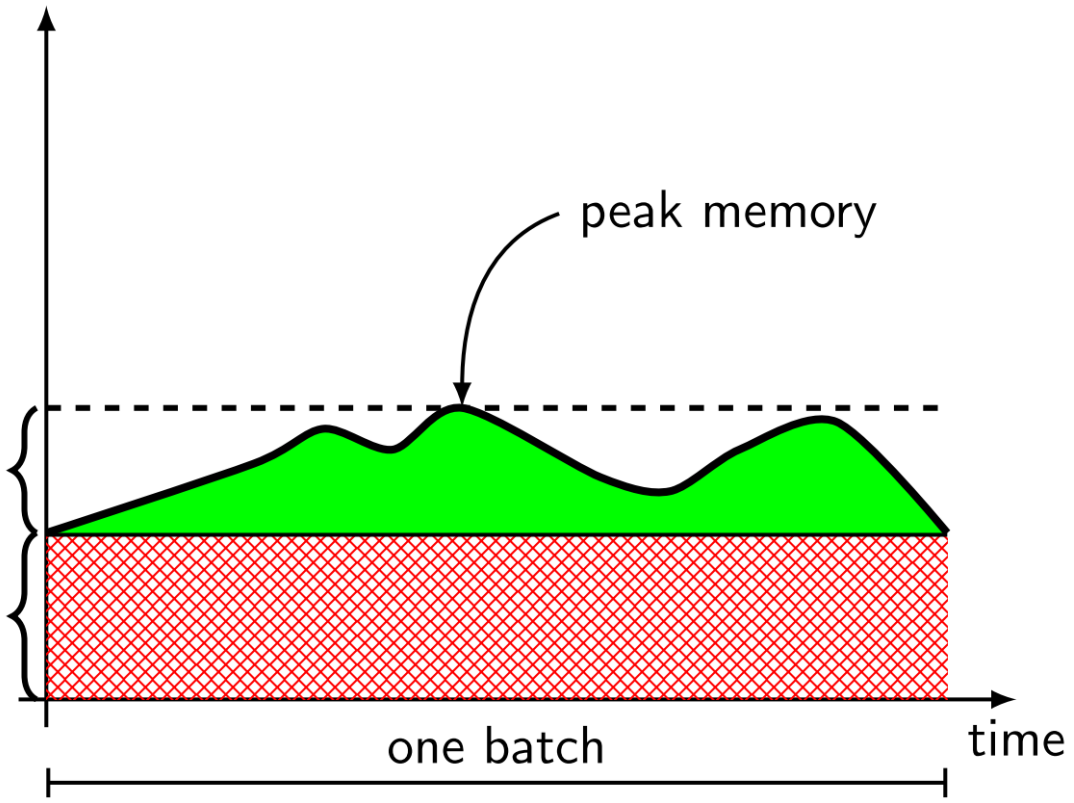
Ahmed Abdourahman Mahamoud
Ingénieur plan de relance Atos/Inria
Equipe DRIM (Eviden)/Equipe TOPAL (INRIA)

I – ROTOR

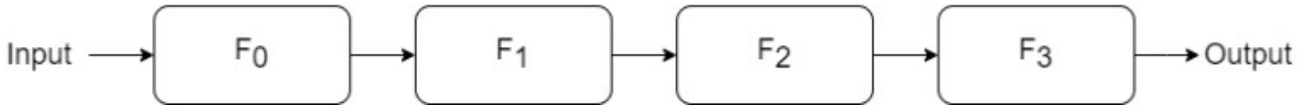
Fonctionnement Normal



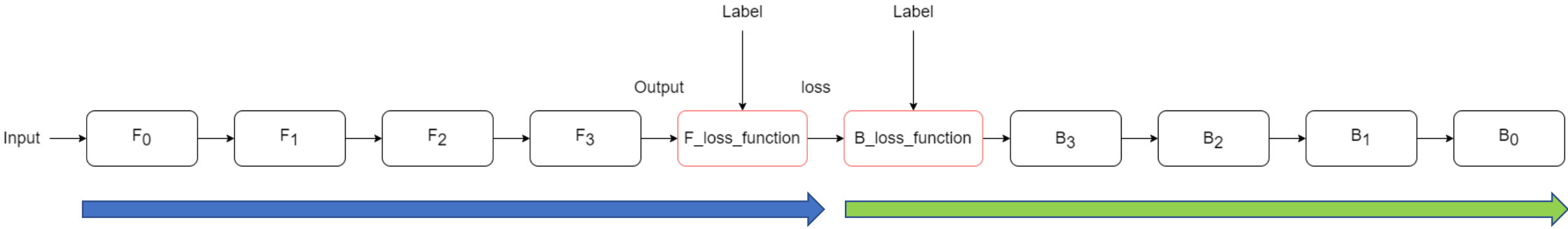
Fonctionnement avec ROTOR



II – Pipe



Modèle simplifié

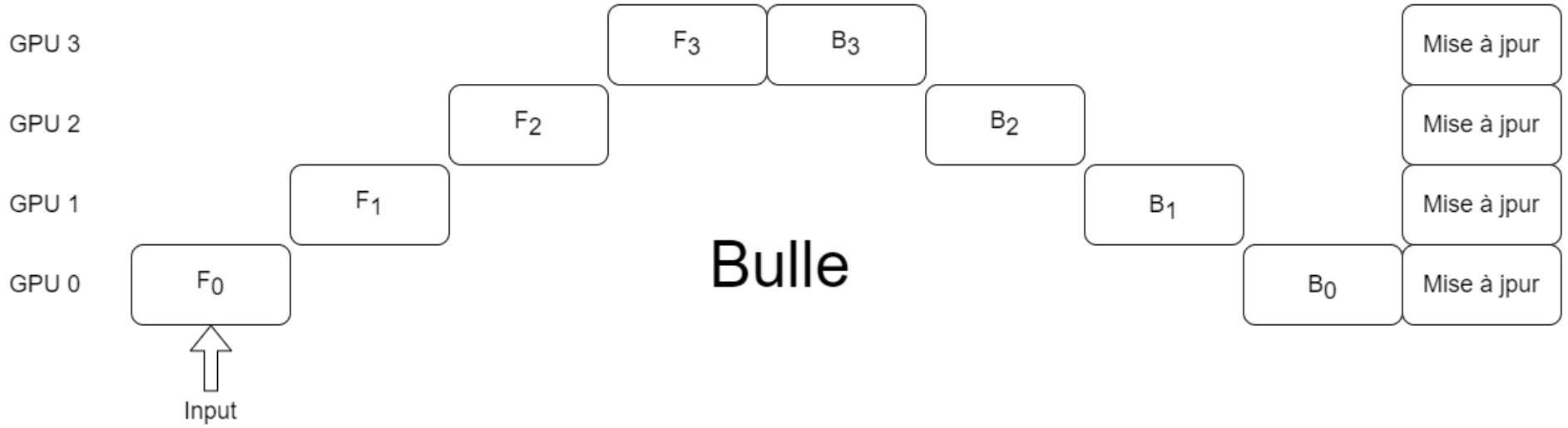


Forward

Backward

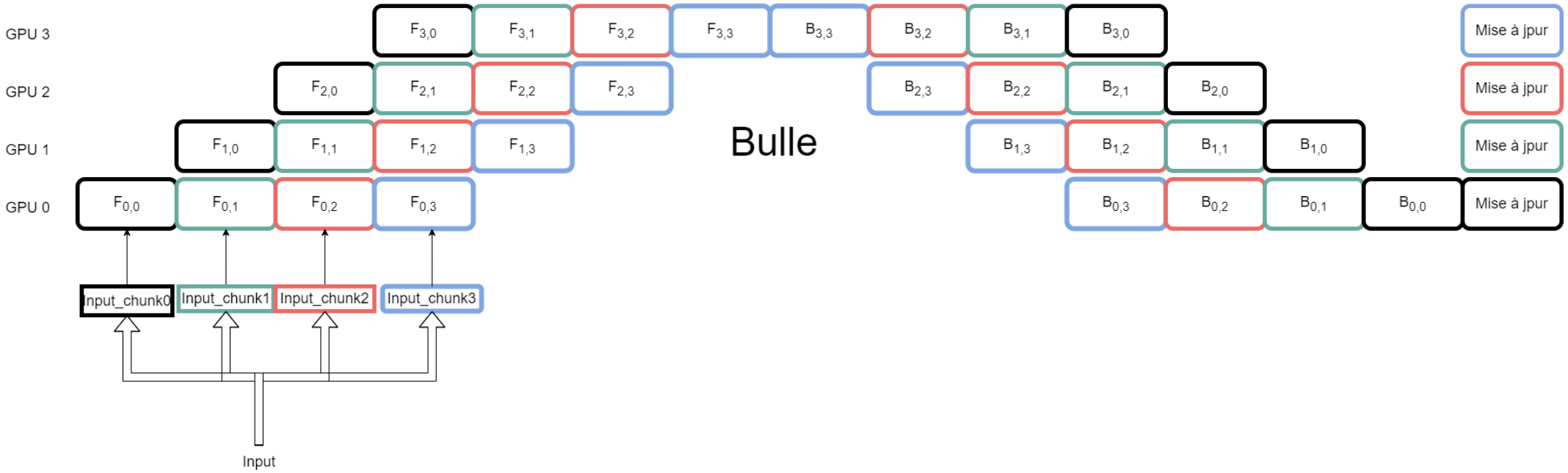
Phases d’entrainement

II – Pipe



Fonctionnement de Pipe

II – Pipe



Fonctionnement de Pipe avec plusieurs chunks

II – Pipe

- Configuration:
  - Modèle : GPT2
  - Architecture : 2 GPUs A100 (40GB/GPU)

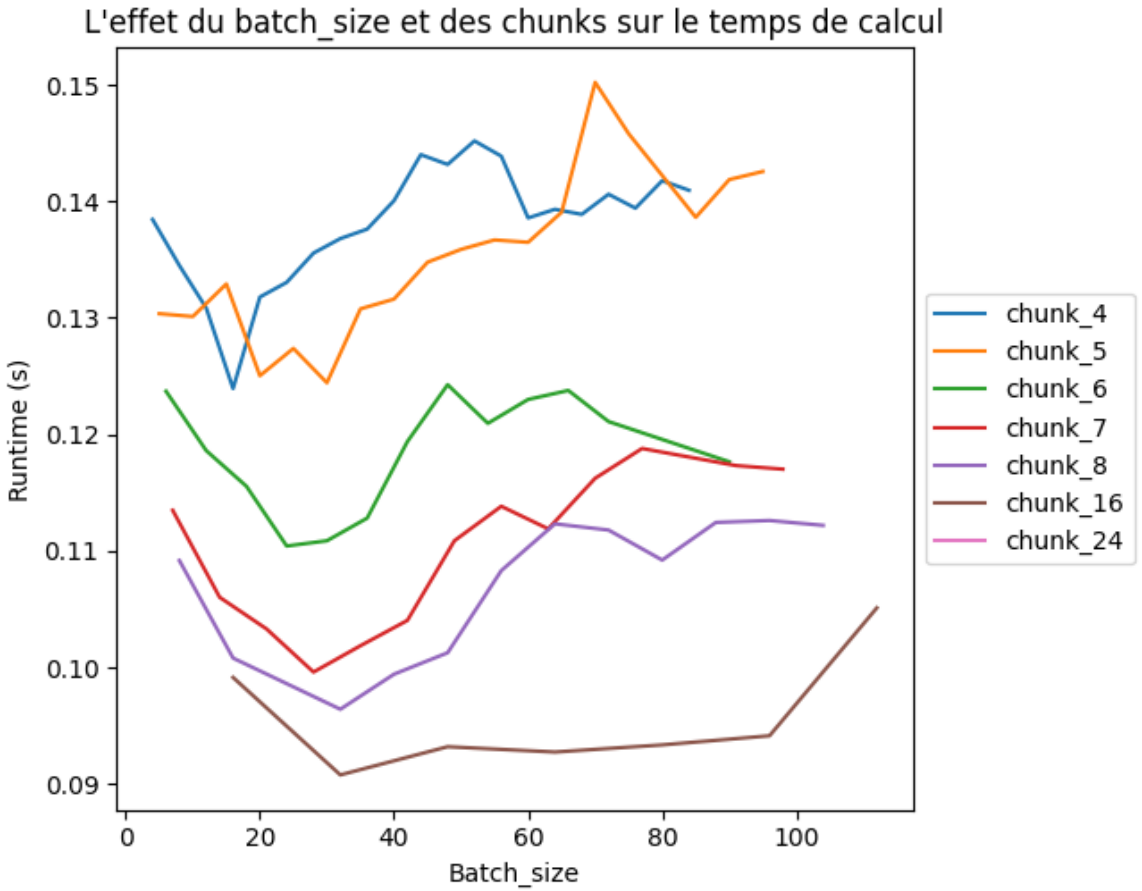
** Batch_size : Taille des données d'entrée*

** Chunk_size = Batch_size/chunk_size : Taille de données par chunk*

*Batch_size	Chunks	*Chunk_size	Temps (s)	Utilisation GPU0	Utilisation GPU1
64	4	16	2,09	49,92%	82,72%
64	8	8	2,3	43,95%	72,52%
64	16	4	3,09	36,58%	59,94%

II – Pipe

Modèle : GPT2 (4,9G)
Architecture : 4 GPUs A100



II – Pipe

Augmentation du batch_size et des chunks :

- Reduction du temps de calcul
- Augmentation de l'utilisation mémoire (Contrôlable avec ROTOR)

Question :

Est-il possible de gagner en temps de calcul en combinant ROTOR avec Pipe?

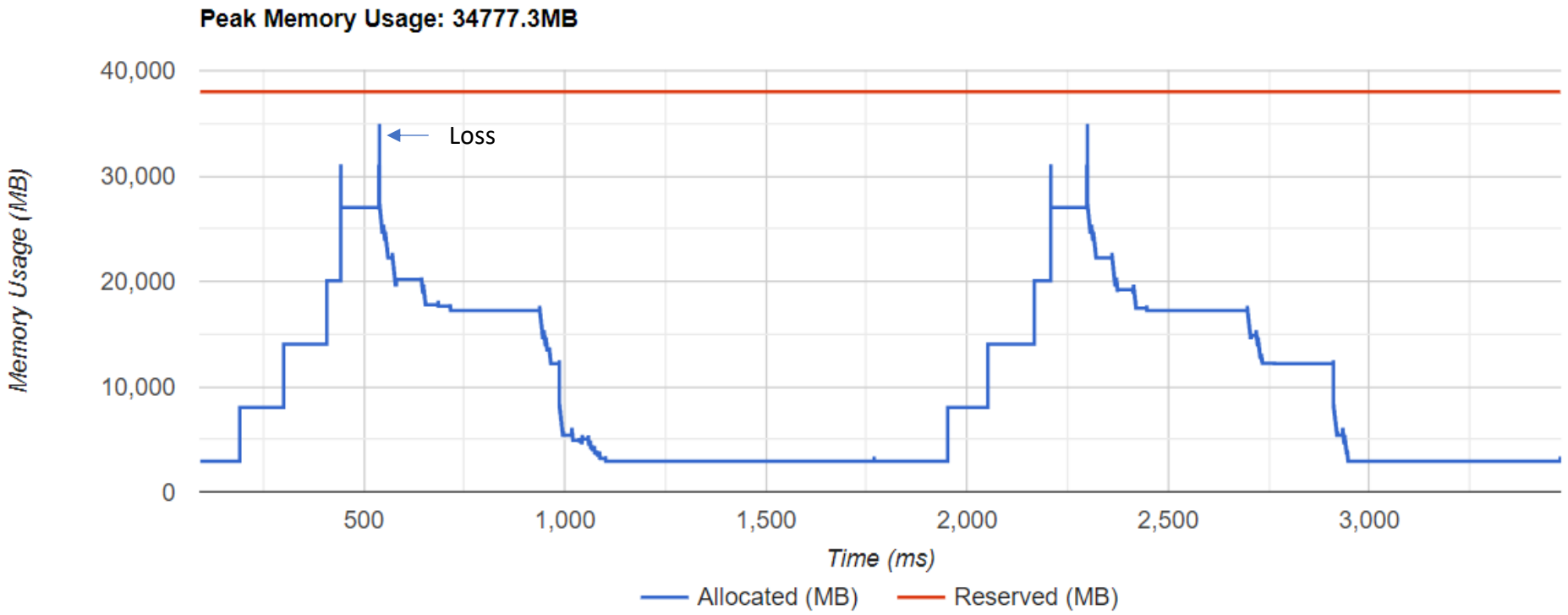
III – Optimisation ROTOR-Pipe

A- Loss fonction

Device
GPU1

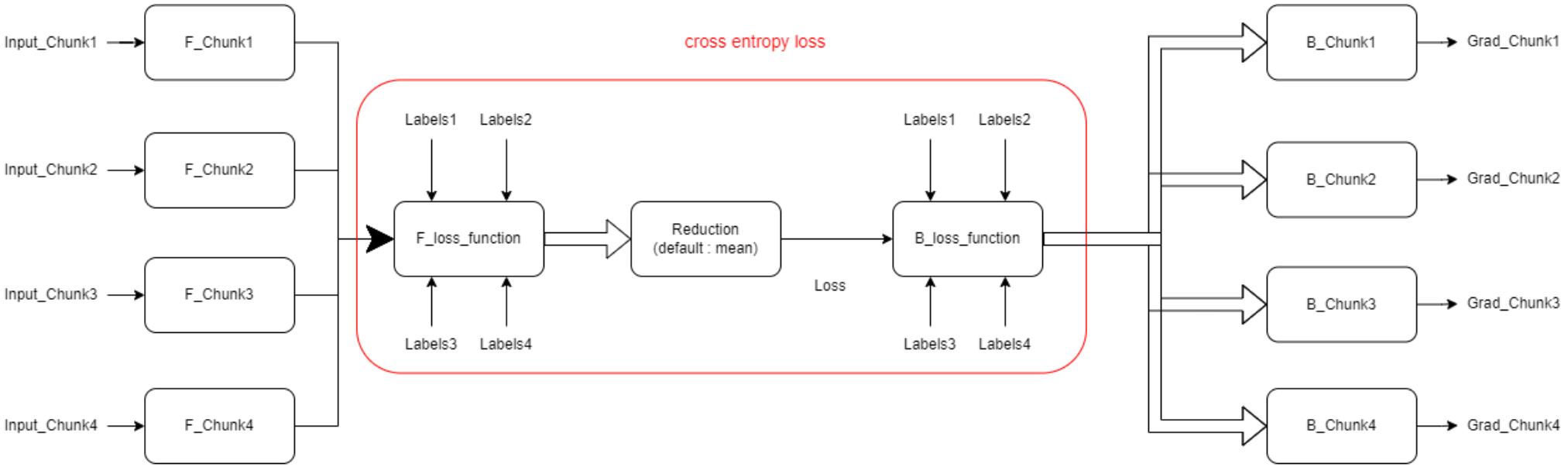
Old loss

Rkgb_pipe_rotor
Batch_size=40,
chunks=4



III – Optimisation ROTOR-Pipe

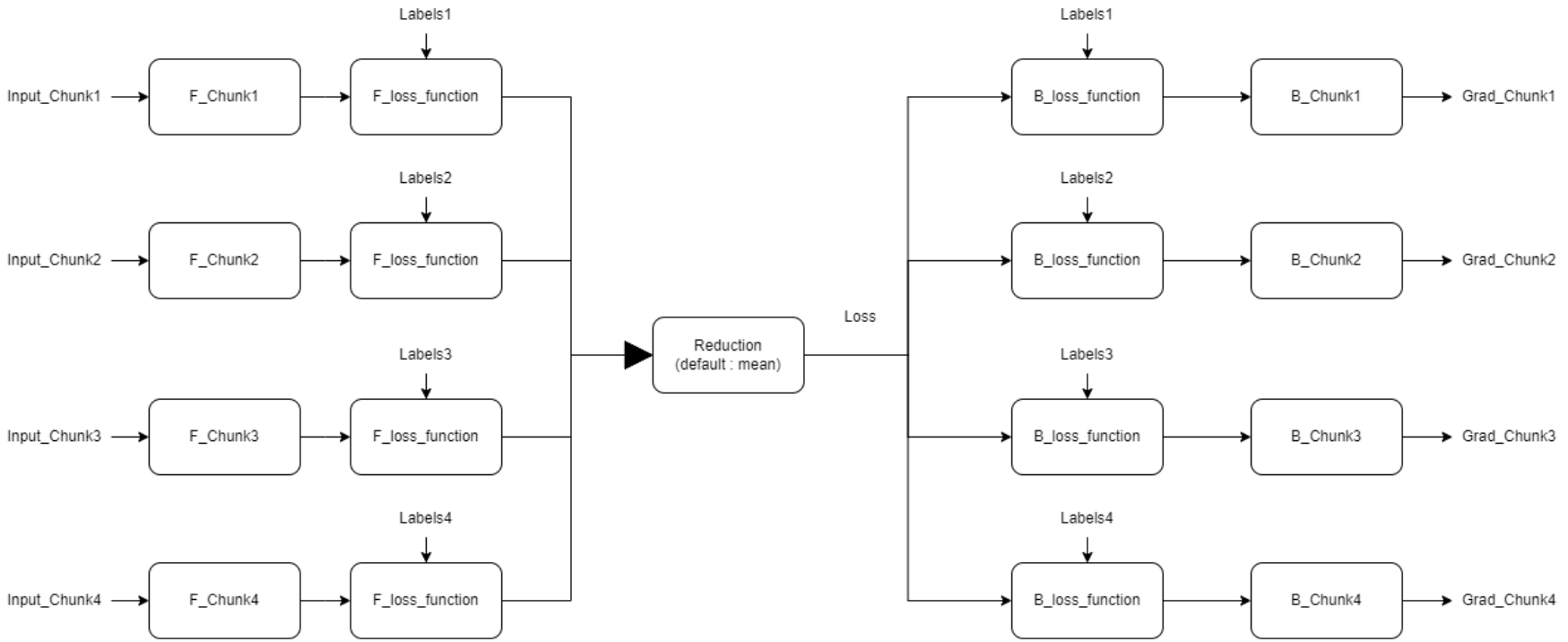
A- Loss fonction



Ancienne organisation de la Loss fonction

III – Optimisation ROTOR-Pipe

A- Loss fonction



Nouvelle organisation de la Loss fonction

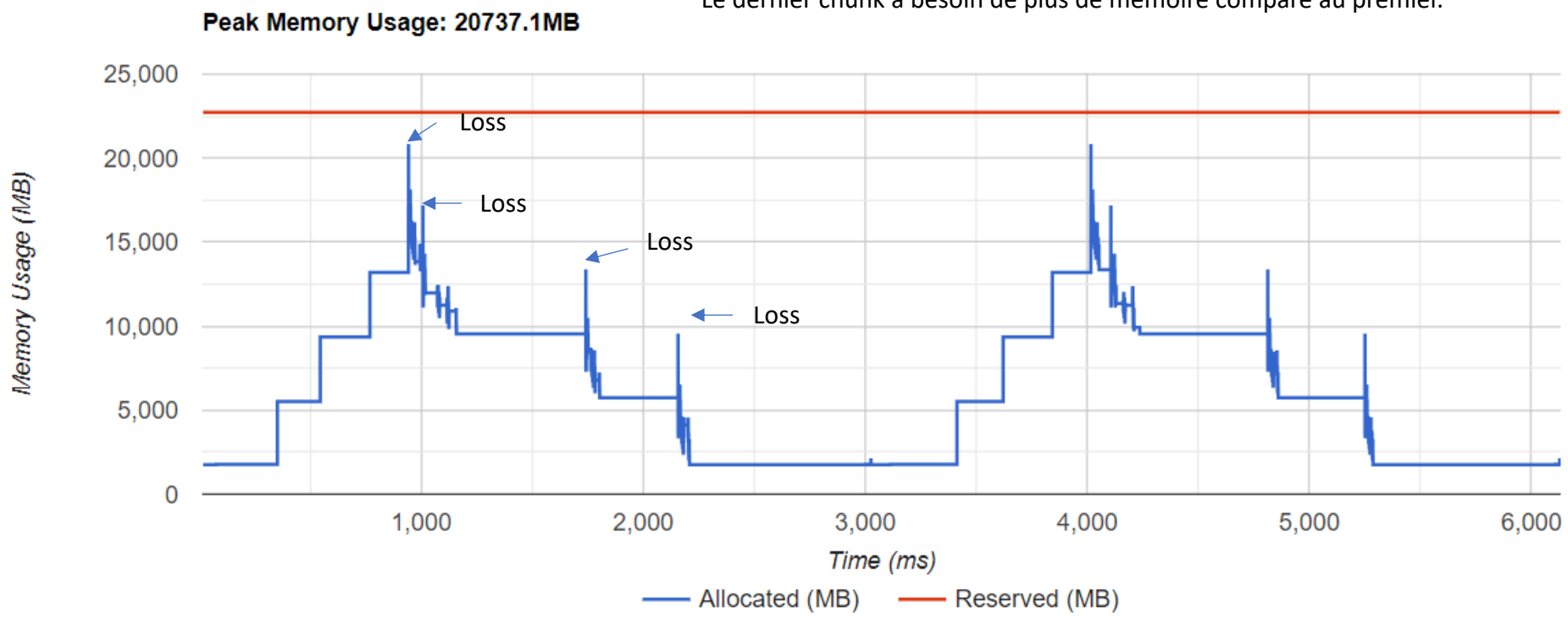
III – Optimisation ROTOR-Pipe

A- Loss fonction

Device
GPU1

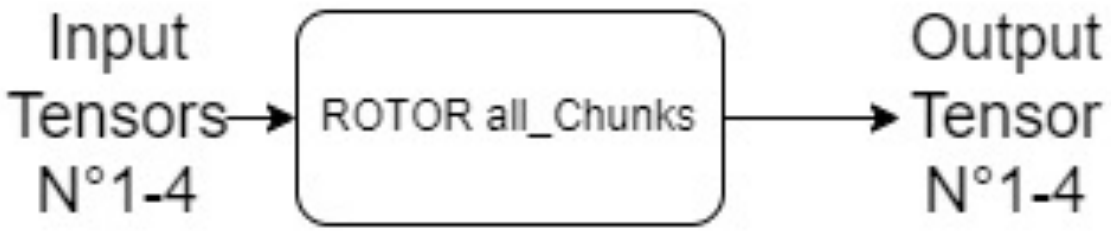
New loss
Rkgb_pipe_rotor
Batch_size=76,
chunks=4

Une utilisation mémoire beaucoup trop faible comparée à la limite fixée rlimit=38,5G. ROTOR suroptimise, car il génère une seule séquence pour tous les chunks.
Le dernier chunk a besoin de plus de mémoire comparé au premier.



III – Optimisation ROTOR-Pipe

B- ROTOR optimisation



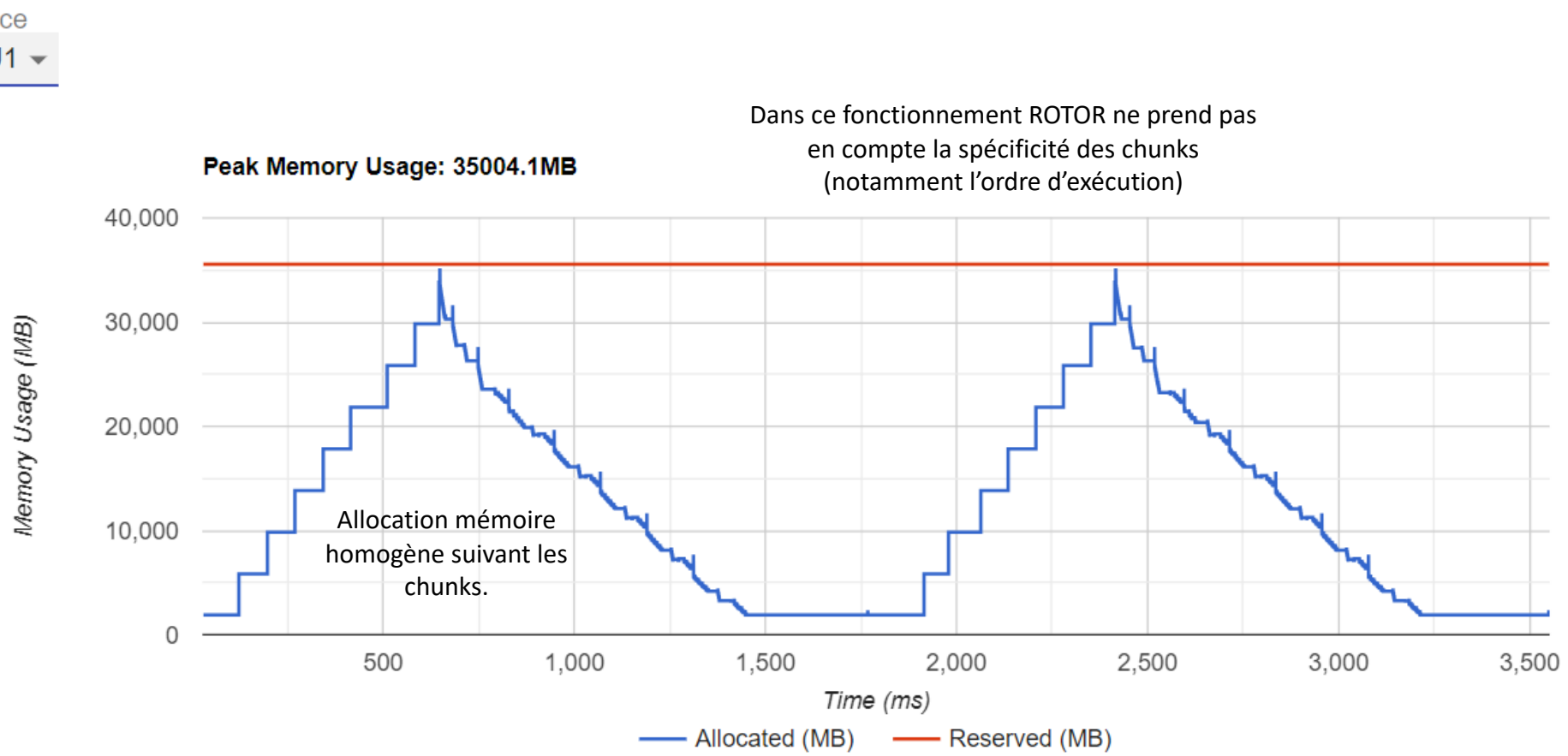
Utilisation de ROTOR avec Pipe

III – Optimisation ROTOR-Pipe

B- ROTOR optimisation

Gpt_seq_pipe_rotor

Batch_size=48, chunks=8

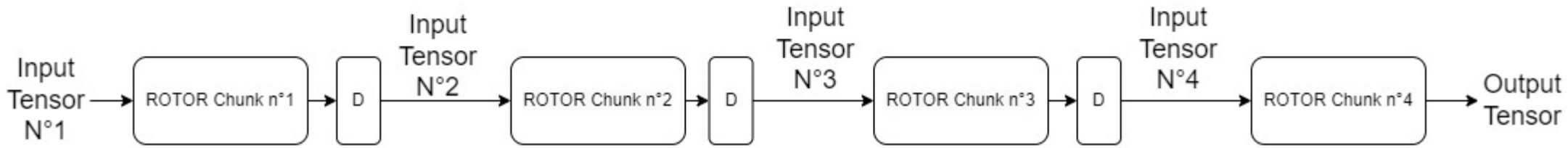


Dans ce fonctionnement ROTOR ne prend pas en compte la spécificité des chunks (notamment l'ordre d'exécution)

Allocation mémoire homogène suivant les chunks.

III – Optimisation ROTOR-Pipe

B- ROTOR optimisation



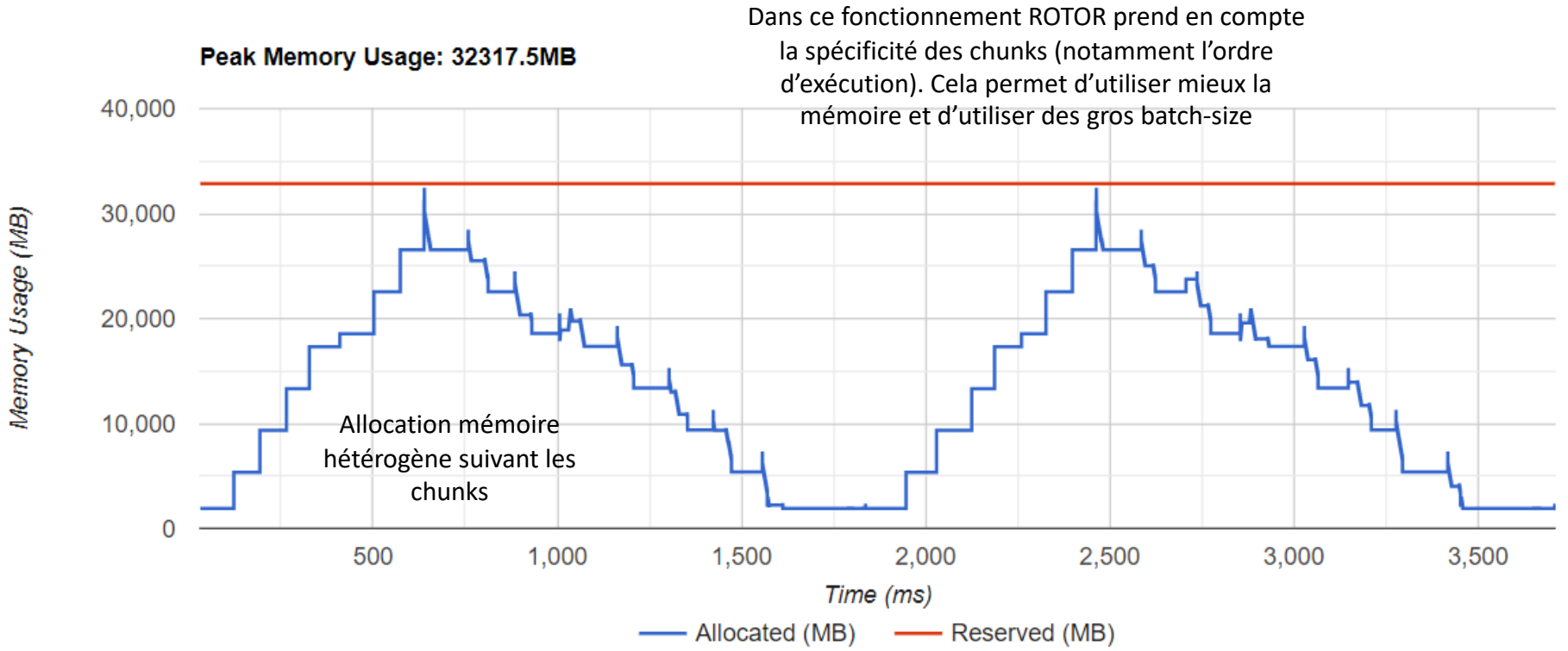
Fonctionnement de ROTOR après modification

III – Optimisation ROTOR-Pipe

B- ROTOR optimisation

Device
GPU1

Gpt_seq_pipe_rotor_modified
Batch_size=48, chunks=8



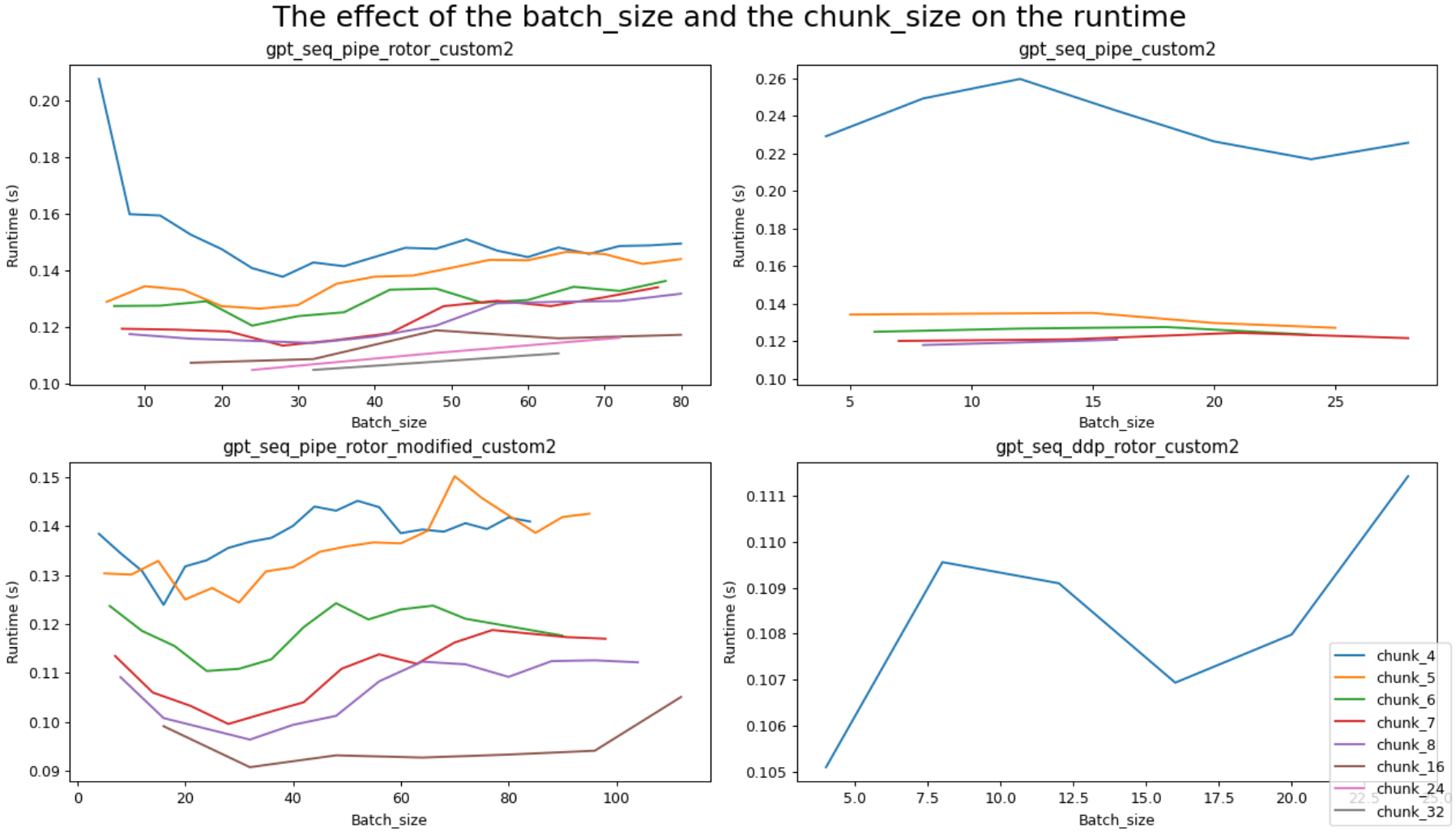
Dans ce fonctionnement ROTOR prend en compte la spécificité des chunks (notamment l'ordre d'exécution). Cela permet d'utiliser mieux la mémoire et d'utiliser des gros batch-size

Allocation mémoire hétérogène suivant les chunks

III – Optimisation ROTOR-Pipe

C - Résultats

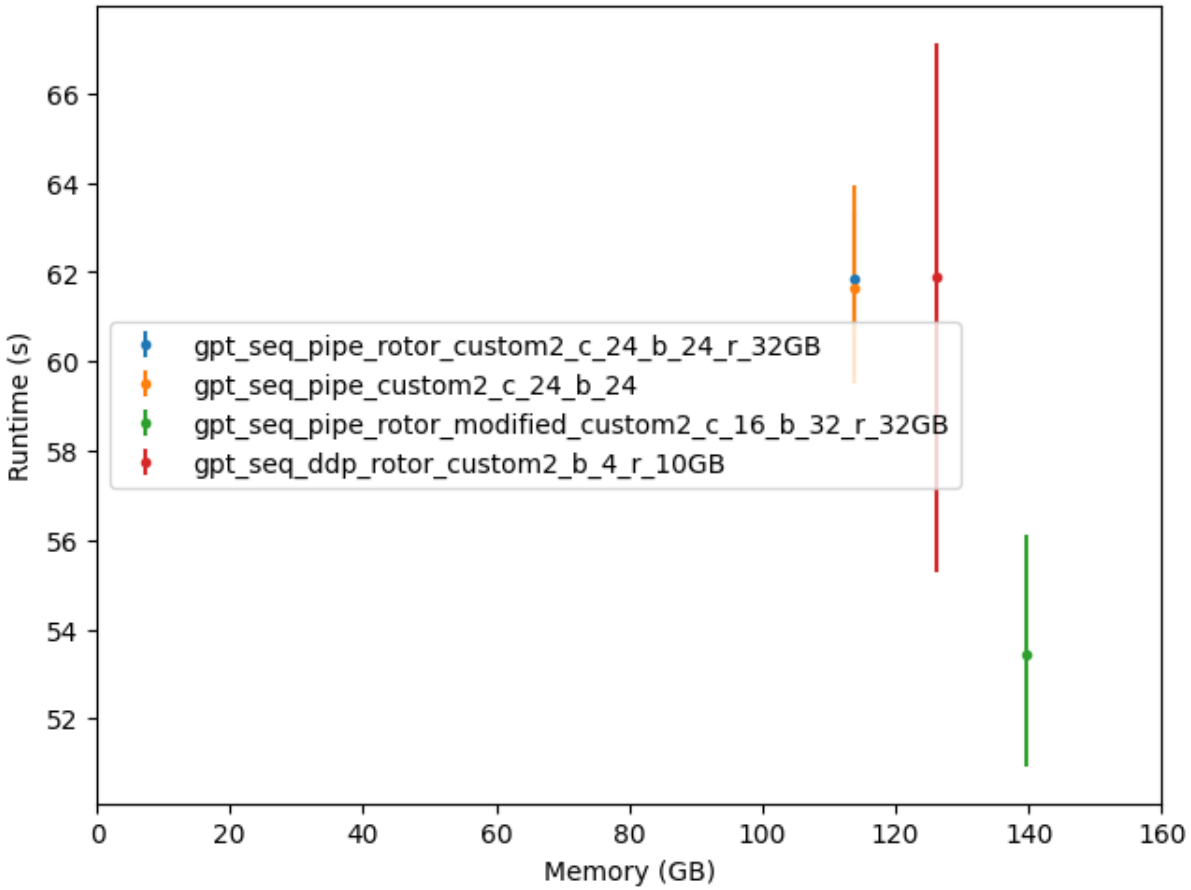
Modèle : GPT2 (4,9G)
Architecture : 4 GPUs A100



III – Optimisation ROTOR-Pipe

C - Résultats

Modèle : GPT2 (4,9G)
Architecture : 4 GPUs A100



IV – Conclusion

Suite des travaux :

- Finaliser les Tests
- Optimisation de Pipe