Safe Optimized Static Memory Allocation for Parallel Deep Learning

Ioannis Lamprou[1] Zhen Zhang[1] Javier de Juan[1] Hang Yang[1]
Yongqiang Lai[2] Etienne Filhol[1] Cedric Bastoul[1]

[1]Huawei Technologies France [2]Huawei Technologies China

MLSys 2023, June 5th, Miami Beach, FL, USA

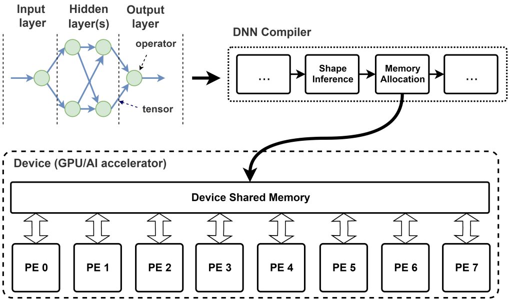
Outline

Memory for Deep Neural Nets (DNNs)

Why Care?

- Large-scale era: deeper and wider neural networks
- Potent AI accelerators, yet with limited memory
- Fit whole model onto fewer devices

Static Execution for Parallel



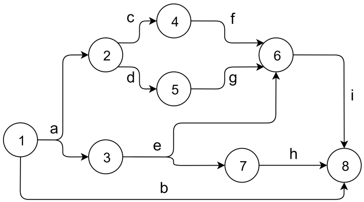
Benefits

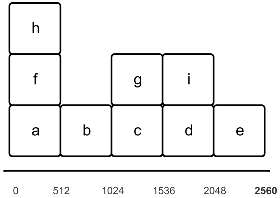
- Compile on host, then load and execute on device
- Avoid OOM, fragmentation, reallocation, relaunching
- Tune the parallelism strategy for large models!

Outline

From Offset Calculation ...

MXNet [Chen et al., 2015]
Chainer [Sekiyama et al., 2018]
TF Lite [Lee et al., 2019, Pisarchyk and Lee, 2020]



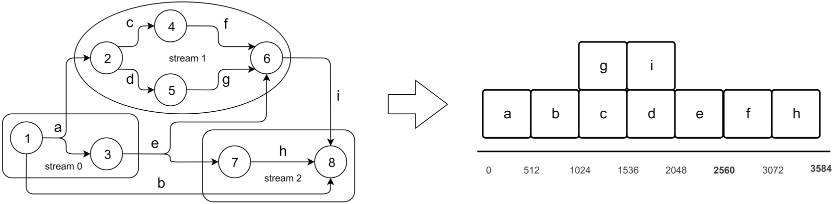


$L_a = [1, 3], L_f = [4, 6]$, therefore a and f safe to overlap

Definition 1 (Offset Calculation).

Given a topologically sorted DNN, return a start **offset for each tensor**, such that no two tensors t_1, t_2, where $L_{t_1} \cap L_{t_2} \neq \emptyset$, overlap in memory and the total footprint is minimized.

... to Offset Calculation for Parallel



Topological sorting valid only within stream: *a* and *f* unsafe to overlap

Definition 2 (Offset Calculation for Parallel).

Given a **multi-stream** DNN, return a start offset for each tensor, so that no two tensors overlap, if they might be needed simultaneously in memory, and the total footprint is minimized.

Offset Calculation for Parallel

Challenges

- Global lifetime cannot determine safe reuse
- Time complexity $\downarrow$ to enable parallel strategy tuning
- Capture general parallelism scenario

Contributions

- Fast computing of provably safe memory reuse constraints
- Fast offset assignment, while (nearly) optimal footprint
- Validation in open-source framework MindSpore (SOMAS)

Outline

The Problem

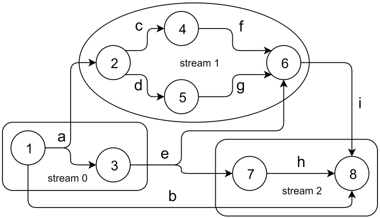
Definition 3 (Safe Pair).

An unordered pair of tensors $\{t_1, t_2\}$ is called a **safe pair** if there is no need to maintain t_1 and t_2 concurrently in memory for **any** potentially realized parallel execution of the DNN.

Definition 4 (Multi-Stream Safety).

Given a multi-stream DNN, for each pair of tensors $\{t_1, t_2\}$ decide whether $\{t_1, t_2\}$ is a safe pair.

Graph-based



- $DestNodes[a] = \{2, 3\}$
- $AncNodes[source(f)] = \{1, 2\}$
- $a \notin AncTensors[f]$

- Computational graph $G = (N, A)$ and tensor set T
- $AncNodes[n] := \{n' \in N \mid \text{ there is a path from } n' \text{ to } n\}$
- $DestNodes[t] := \{n'' \in N \mid n'' \text{ receives tensor } t\}$

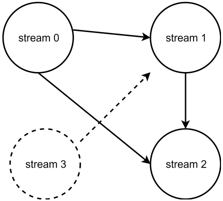
$$AncTensors[t] := \{t' \in T \mid DestNodes[t'] \subseteq AncNodes[source(t)]\}$$

$$\{t_1, t_2\} \text{ safe pair if } t_1 \in AncTensors[t_2] \text{ or } t_2 \in AncTensors[t_1]$$

Stream-based

Input: A DNN stream set S and tensor set T.
Output: A set $U'' \subseteq \binom{T}{2}$ of unsafe pairs.

$U \leftarrow \binom{T}{2}$;
$U' \leftarrow \mathit{AncestorStreamsReuse}(S, T, U)$;
$U'' \leftarrow \mathit{SameStreamReuse}(S, T, U')$;
return U'';



Idea: Stream graph

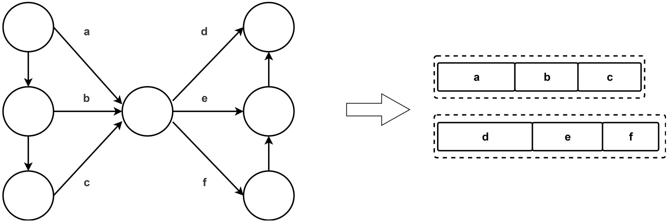
- Pairs of tensors in unrelated streams unsafe by default
- Only check safe pairs for sources in ancestor/same stream
- $\mathit{DestStreams}[t'] \subseteq \mathit{AncStreams}[\mathit{stream}(t)]$

Theorem 5.

Stream-based solves Multi-Stream Safety

Outline

Contiguous Constraints



- Set of Contiguous Constraints $\{C_1, C_2, \ldots, , C_l\}$
- $C_i = [t_{i,1}, t_{i,2}, \ldots, t_{i,k_i}]$
- $offset(t_{i,j}) = offset(t_{i,j-1}) + size(t_{i,j-1})$ for all $j = 1, 2, \ldots k_i$

- Tensor concatenation may not be possible (tensor shapes)
- “Union” of safe pairs may overprotect (5% in ResNet50)

The Problem

Definition 6 (Offset Assignment for Parallel).

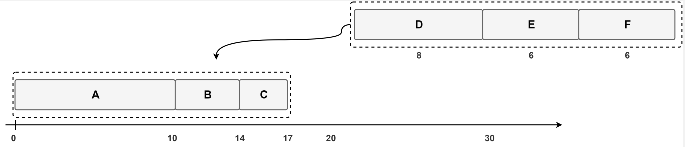
Given a set of tensors, a set of unsafe pairs and contiguous constraints, return a start offset for each tensor so that

- any two tensor offset intervals do not overlap if unsafe
- all contiguous constraints are respected, and
- the total footprint is minimized.

Key Concepts

Algorithm Design

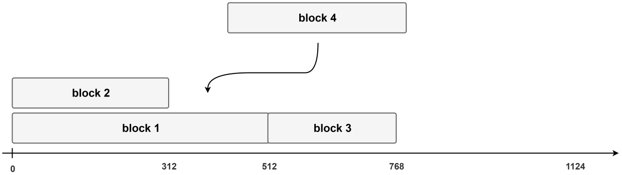
1. Sort **blocks** of tensor(s) according to some criteria
2. Determine forbidden offset intervals for current block

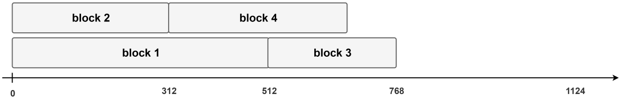


Special care for contiguous: if $\{E, B\}$ unsafe, E unsafe start on $[5, 14]$, so D unsafe start on $[-3, 6]$, i.e., on $[0, 6]$

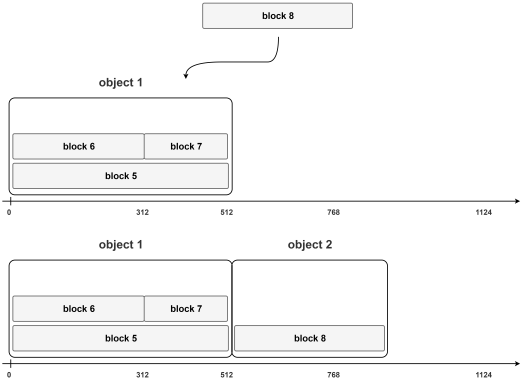
3. Decide offset interval for current block out of safe ones

From Single Object ...





... to Many Objects



Iterate steps 2,3 within each object until placement, do not examine whole space.
Break if unsafe with object-spanning block. If no placement possible, create new object.

Outline

Multi-Stream Safety

Multi-Stream Safety tested in MindSpore on Ascend 910 (solving time in milliseconds)

Network	Graph Based	Stream Based	Speedup
BERT-base	957	620	~35%
BERT-large	4043	2289	~43%
BERT-nezha	5275	2959	~44%
FaceRecognition	1376	845	~39%
PanGu-α (2.6B)	13845	10359	~25%
ResNet-50	32	20	~38%
Tiny-BERT	143	96	~33%
FaceDetection	693	546	~21%
Transformer	720	568	~21%
MobileNetv2	57	42	~26%

Offset Assignment I

Training experiments: peak memory in GB, solving time (milliseconds) in italic

	BERT-base	BERT-large	BERT-nezha	FaceRecognition	PanGu-α (2.6B)
Memory Usage					
Naïve Allocation	42.7816	83.3553	61.5739	77.6916	1349.1400
Single Object (SO)	13.5119	24.9171	14.7778	15.7456	18.4541
Many Objects (MO)	13.5121	24.9172	14.7854	15.7797	18.4541
Lower Bound (LB)	13.5119	24.9171	14.6860	15.7456	18.4541
Memory Error					
MO to SO	0.00148%	0.00040%	0.05143%	0.21656%	0%
min(SO,MO) to LB	0%	0%	0.62509%	0%	0%
Solving Time					
Single Object (SO)	*600*	*3596*	*4161*	*2925*	*15478*
Many Objects (MO)	*316*	*2090*	*2185*	*869*	*12586*
MO to SO gain	~47%	~42%	~48%	~70%	~19%

Offset Assignment II

Training experiments: peak memory in GB, solving time (milliseconds) in italic

	ResNet-50	Tiny-BERT	FaceDetection	Transformer	MobileNetv2
Memory Usage					
Naïve Allocation	3.3598	5.17475	13.5162	34.2267	64.1832
Single Object (SO)	1.4133	0.70180	3.19949	7.54506	17.6506
Many Objects (MO)	1.4132	0.69726	3.20942	7.54506	17.6662
Lower Bound (LB)	1.4056	0.68938	3.19949	7.54506	17.6423
Memory Error					
MO to SO	**-0.00708**%	**-0.64690**%	0.31036%	0%	0.08838%
min(SO,MO) to LB	0.54069%	1.14306%	0%	0%	0.04705%
Solving Time					
Single Object (SO)	*49*	*101*	*808*	*317*	*611*
Many Objects (MO)	*29*	*44*	*424*	*152*	*158*
MO to SO gain	~41%	~56%	~48%	~52%	~74%

Large model with Contiguous Constraints

Training experiment: PanGu-α large model (400-700 contiguous constraints)

	PanGu-α (8B)	PanGu-α (13B)
Baseline (MindSpore before our solution)	27.36	31.72
Our Best Result	14.76	25.08
Lower Bound	14.68	24.95
Memory Error		
Our result to Baseline	-46.05%	-20.92%
Our result to Lower Bound	0.54%	0.55%

Conclusion

Recap

- Enable generalized static parallel deep learning
- Safe pairs determining for Multi-Stream Safety
- Many Objects (with contiguous) for Offset Assignment

Future Work

- Choice of multi-streaming
- Global/local topological sorting

Conclusion

Recap

- Enable generalized static parallel deep learning
- Safe pairs determining for Multi-Stream Safety
- Many Objects (with contiguous) for Offset Assignment

Future Work

- Choice of multi-streaming
- Global/local topological sorting

Thank you!

References

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Juhyun Lee, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-device neural net inference with mobile gpus. *arXiv preprint arXiv:1907.01989*, 2019.

Yury Pisarchyk and Juhyun Lee. Efficient memory management for deep neural net inference. *arXiv preprint arXiv:2001.03288*, 2020.

Taro Sekiyama, Takashi Imamichi, Haruki Imai, and Rudy Raymond. Profile-guided memory optimization for deep neural networks. *arXiv preprint arXiv:1804.10001*, 2018.