AI编译器-系列之前端优化

内存分配

ZOMI

Talk Overview of Frontend Optimizer

I. AI 编译器前端优化

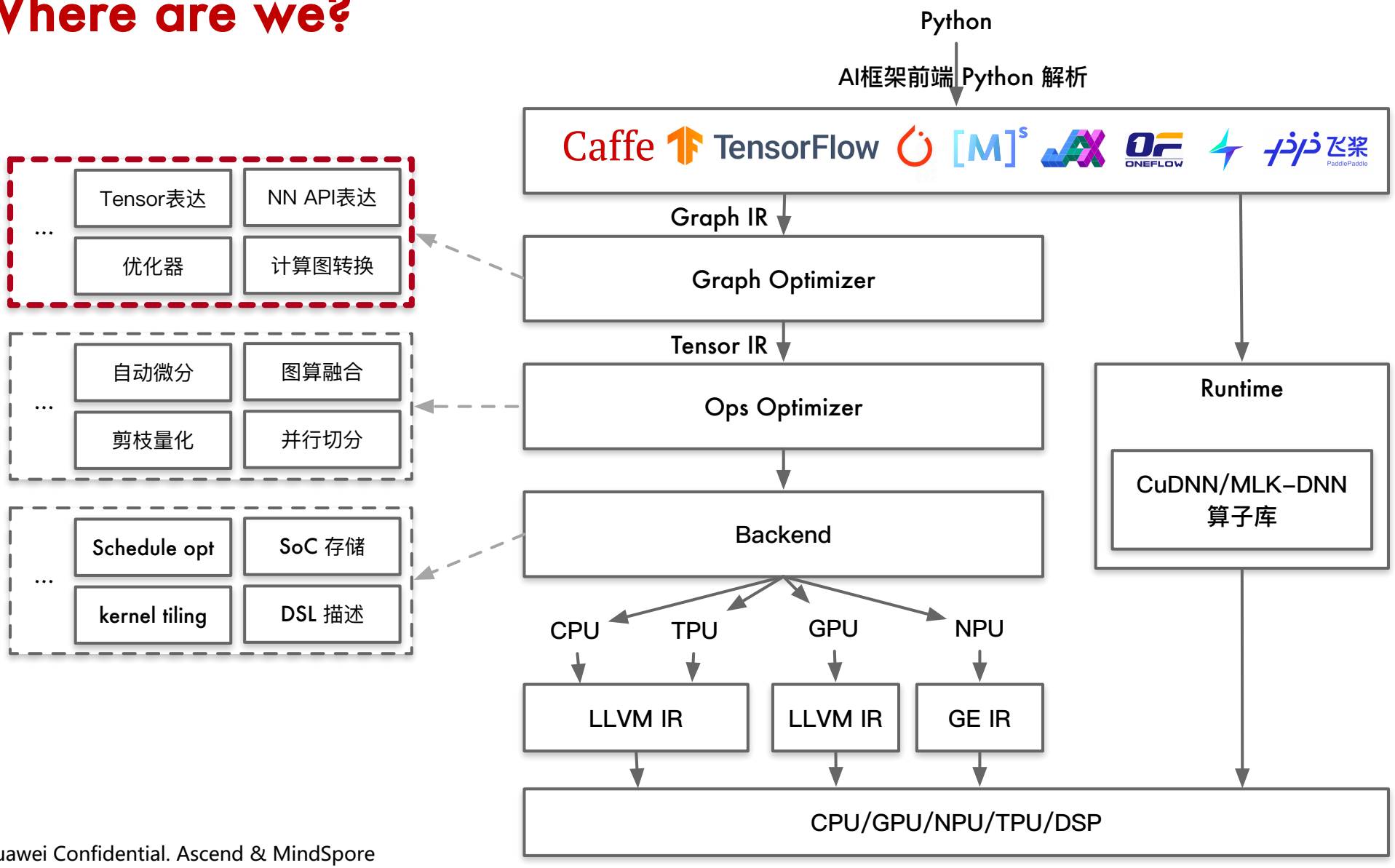
- 图层 - Graph IR
- 算子融合 - OP Fusion
- 布局转换 - Layout Transform
- 内存分配 - Memory Allocation
- 常量折叠 – Constant Fold
- 公共子表达式消除 - CSE
- 死代码消除 - DCE
- 代数简化 - ARM

Talk Overview

memory allocation – 内存分配

- 模型和硬件的内存演进
- 内存的划分与复用好处
- 节省内存的算法

Where are we?



模型对内存需求的和硬件发展

模型需求

模型训练时需要大量的显存空间：

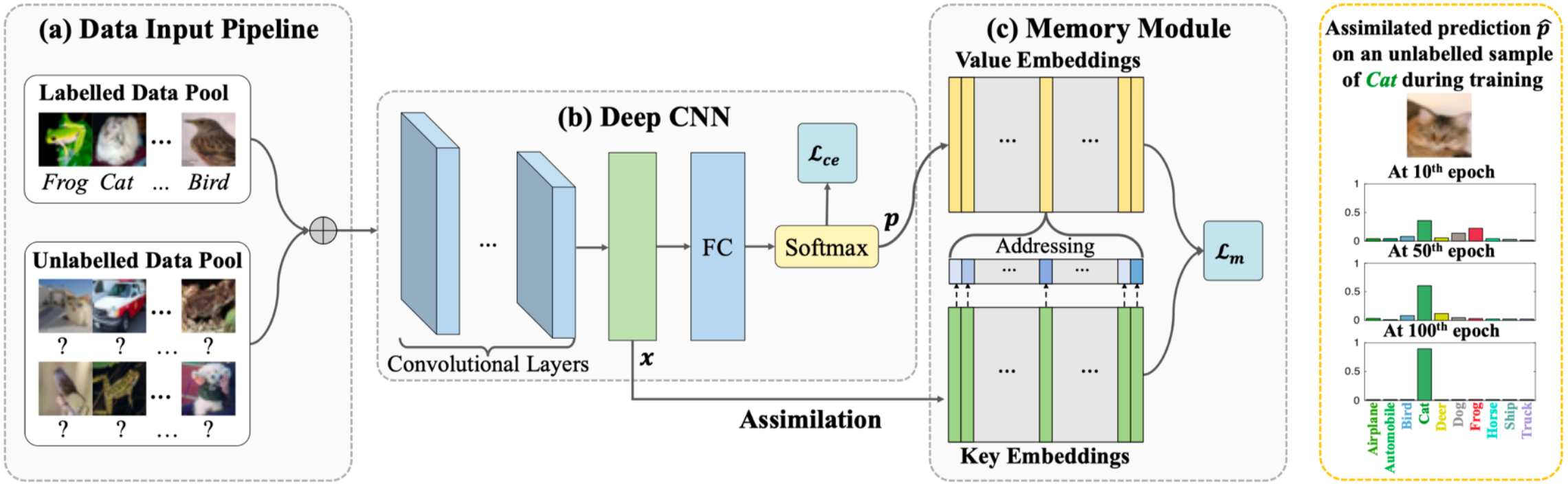
- InceptionV4设置batch size为32训练 ImageNet需要 40GB显存空间 ；
- BERT拥有768个隐藏层，在Batch size设 置为64时需要73GB的显存空间 ；
- 使用ImageNet训练Wide ResNet-152，并 设置Batch size为64需要显存180GB ；

硬件能力

训练硬件能够提供个内存和带宽：

	Tesla K80	Tesla P40	Tesla V100	Tesla T4/T4S	Tesla V100S	Tesla A100
CUDA核心数	2496	3840	5120	2560	5120	6912
单精度浮点性能	8.7 TFLOPS	12 TFLOPS	14 TFOPS	8.1 TFLOPS	16.35 TFLOPS	19.49 TFLOPS
INT8性能	N/A	47 TOPS	N/A	130 TOPS	N/A	N/A
Tensor性能	N/A	N/A	112 TFLOPS	N/A	130 TFLOPS	312 TFLOPS
显存容量	12GB	24GB	16GB	16GB	32GB	80GB
功耗	300W	250W	250W	70W	300W	300W
架构	Kepler	Pascal	Volta	Turing	Volta	Ampere

深度学习训练流程



Question?

- 构建一个n层神经网络会消耗多大的内存空间？

- 假设我们要建设具有n层的神经网络。神经网络的一个典型实现是需要给每一层的输出分配节点的空间，以及反向传播梯度值。这意味着我们需要大致2n个内存单元。这在显式反向图中也一样，在反向传播节点与前向传播节点大致相同。

内存划分与内存复用好处

静态内存

- Parameter - 网络中的权重
- Value Node - 网络中的常量
- Output - 网络的输出

- 比如一些固定的算子在整个计算图中都会使用，此时需要再模型初始化时一次性申请完内存空间，在实际推理时不需要频繁申请操作，提高性能

内存占用情况—Static Memory

Static

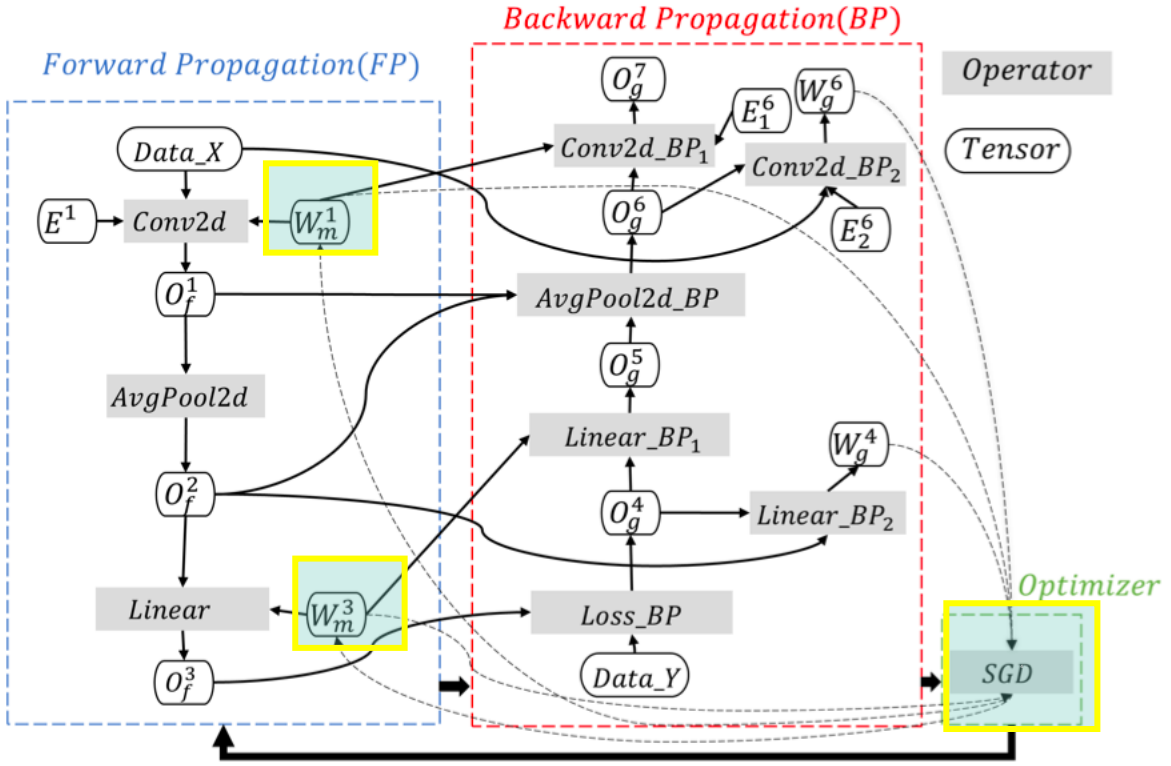


Figure 3: Computation graph for training the DL model in Figure 2. Ovals represent tensors in which W stands for weight tensor, O for In/Out tensor, and E for ephemeral tensor. Rectangles are operators.[2] Dash lines denote weight updates by SGD.

动态内存

- Output Tensor - 网络中的算子的输出Tensor
- Workspace Tensor - 网络中的部分算子在计算过程中的临时buffer

- 动态内存分配：对于中间临时的内存需求，可以进行临时申请和释放，节省内存使用，提高模型并发能力

内存占用情况—Dynamic Memory

Dynamic

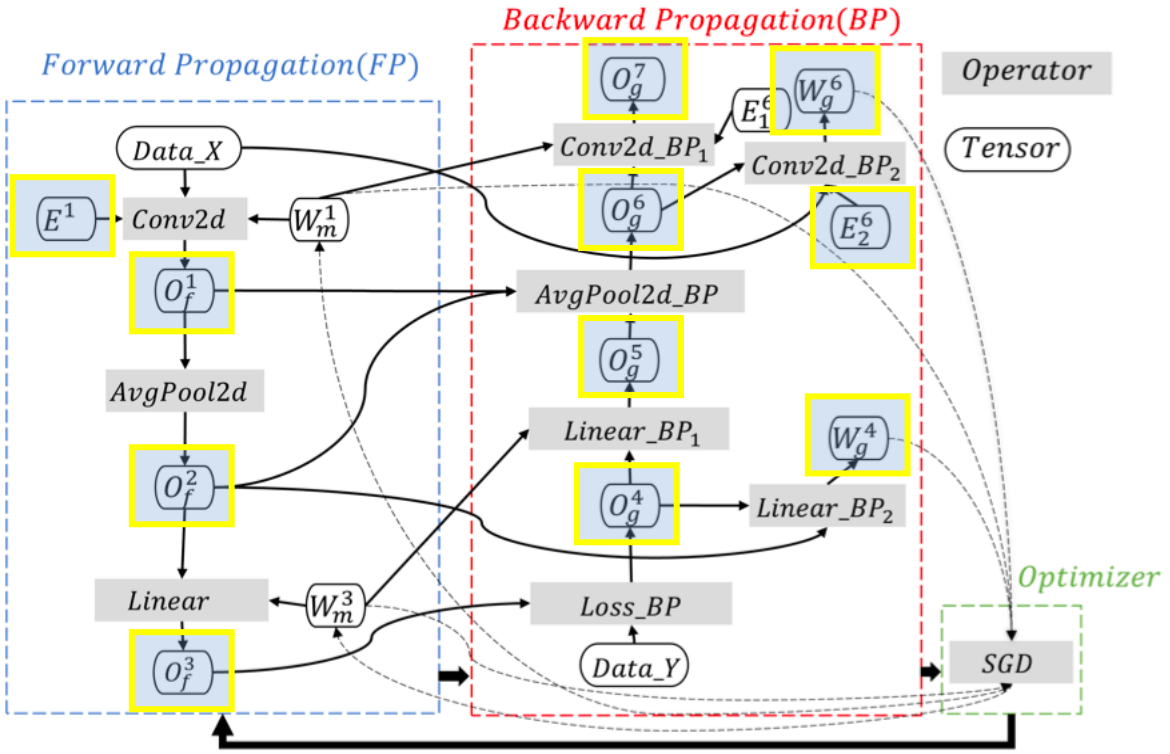
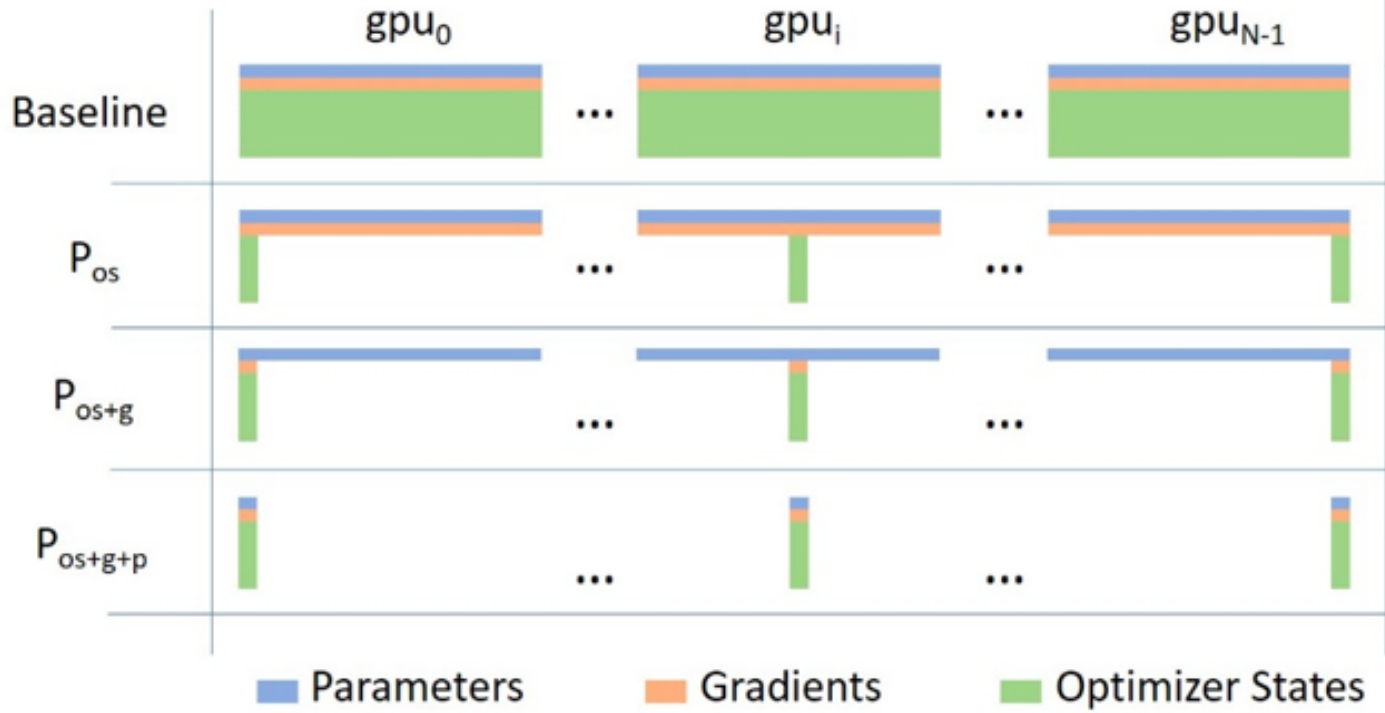


Figure 3: Computation graph for training the DL model in Figure 2. Ovals represent tensors in which W stands for weight tensor, O for In/Out tensor, and E for ephemeral tensor. Rectangles are operators.[2] Dash lines denote weight updates by SGD.

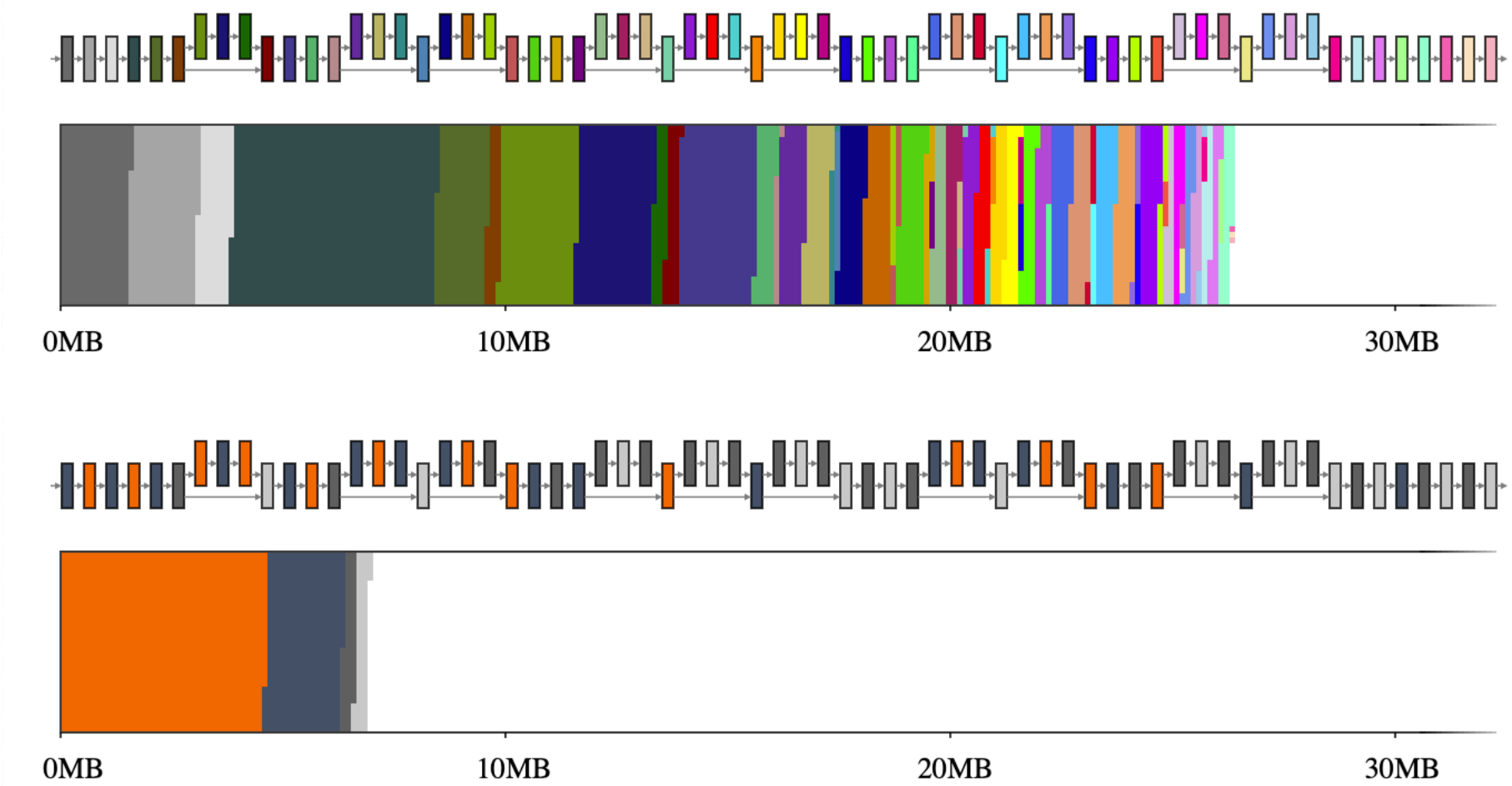
内存情况

- ZeRO: Memory Optimizations Toward Training Trillion Parameter Models



	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

MobileNet v2



https://blog.tensorflow.org/2020/10/optimizing-tensorflow-lite-runtime.html

节省内存的算法

节省内存算法

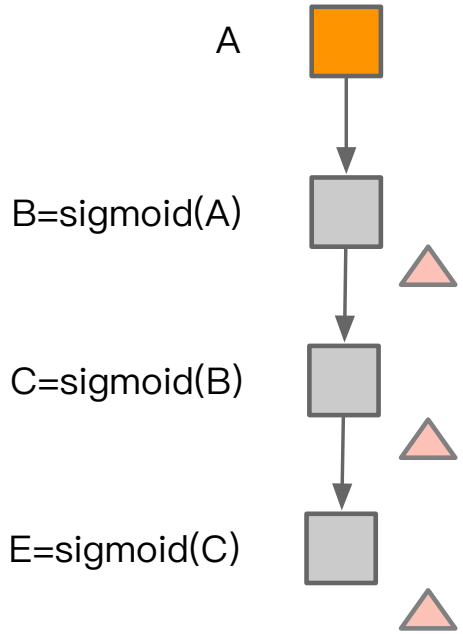
- 空间换内存：如卸载到CPU (CPU Offload)
- 计算换内存： 重计算 (Gradient Checkpointing)
- 模型压缩：如量化训练 Quantification， 剪枝等压缩算法
- 内存复用：利用AI编译器对计算图中的数据流进行分析，以允许重用内存

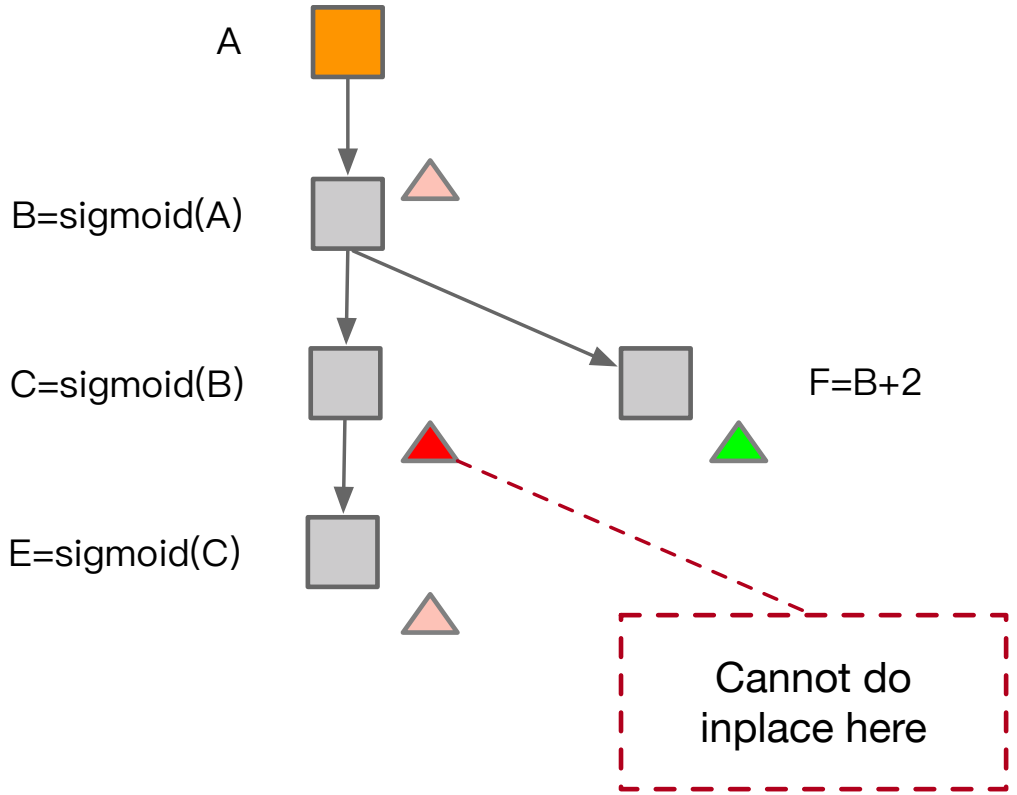
Question?

- 如何在每一个计算图的输出节点上分配内存?

替代操作

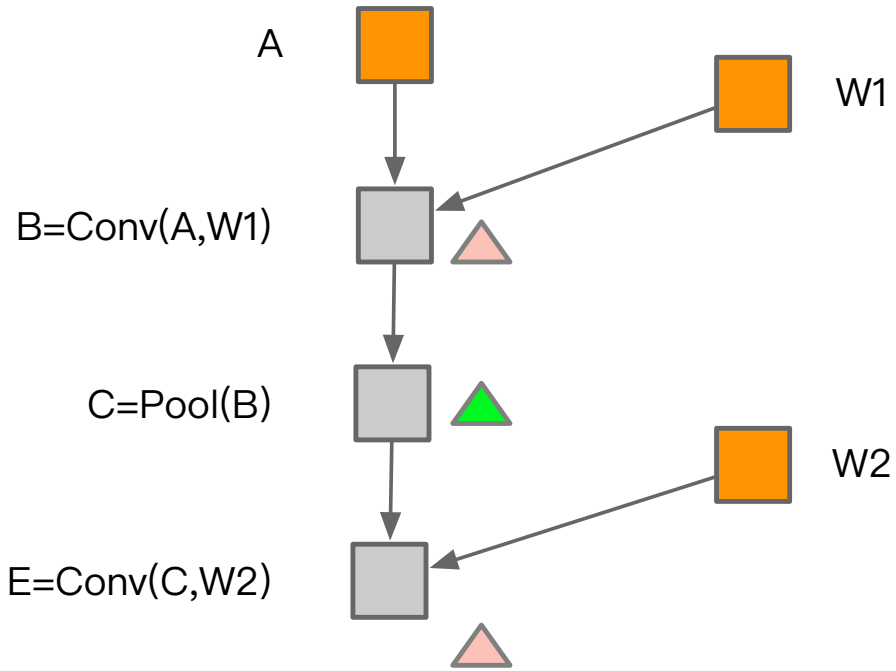
- Inplace operation：如果一块内存不再需要，且下一个操作是element-wise，可以原地覆盖内存





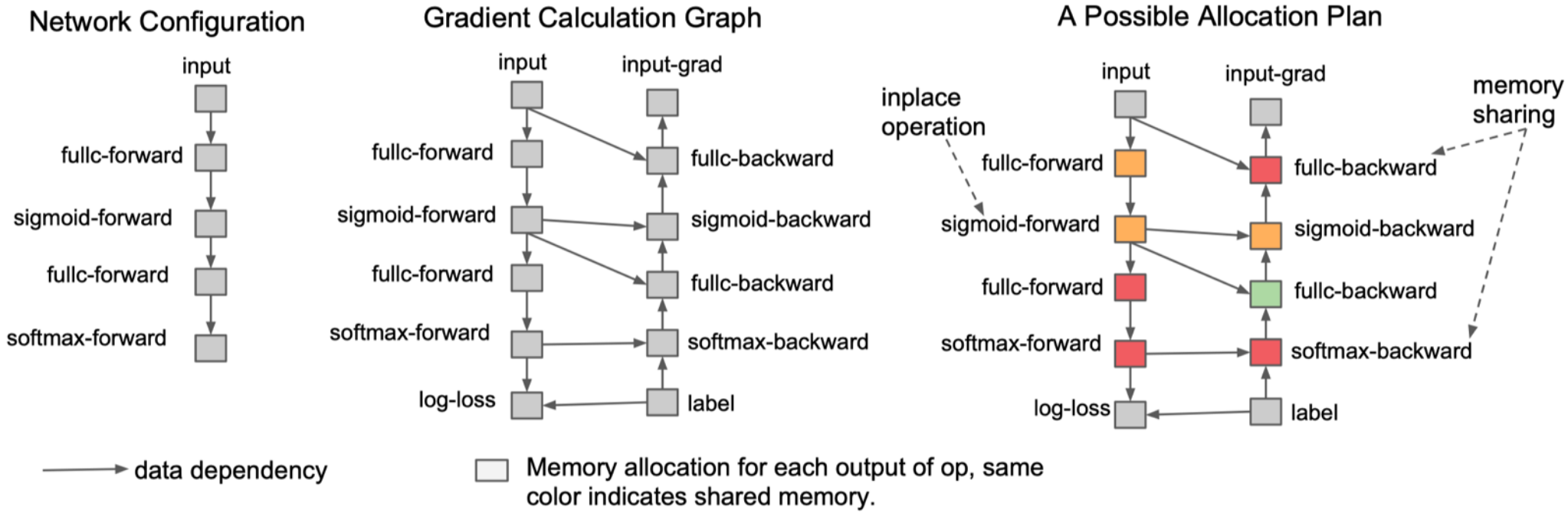
内存共享

- Memory sharing：两个数据使用内存大小相同，且有一个数据参与计算后不再需要，后一个数据可以覆盖前一个数据



内存优化方法

- Inplace operation：如果一块内存不再需要，且下一个操作是element-wise，可以原地覆盖内存
- Memory sharing：两个数据使用内存大小相同，且有一个数据参与计算后不再需要，后一个数据可以覆盖前一个数据

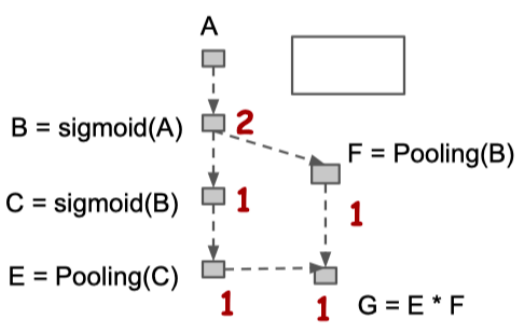


Question?

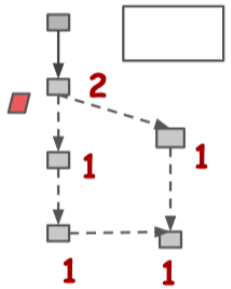
- 怎样才能正确地分配内存？
- 它与传统编译器寄存器分配非常相似，我们可以借鉴很多思想

内存优化方法

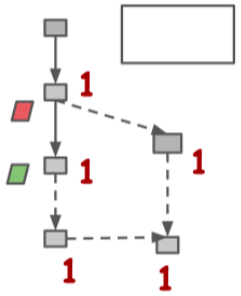
- 对计算图进行分析，找到 Inplace operation 和 Memory sharing：



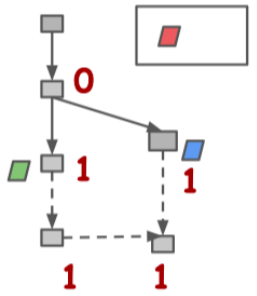
Initial state of allocation algorithm



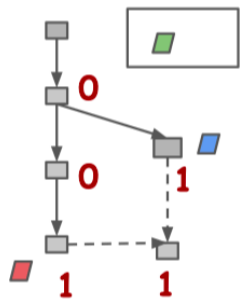
step 1: Allocate tag for B



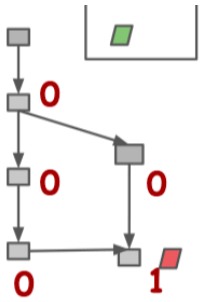
step 2: Allocate tag for C, **cannot do inplace** because B is still alive



step 3: Allocate tag for F, release space of B



step 4: Reuse the tag in the box for E



step 5: Re-use tag of E, This is an **inplace optimization**

Final Memory Plan

internal arrays, same color indicates shared memory.

count ref counter on dependent operations that yet to be full-filled

data dependency, operation completed

data dependency, operation not completed

Tag used to indicate memory sharing on allocation Algorithm.

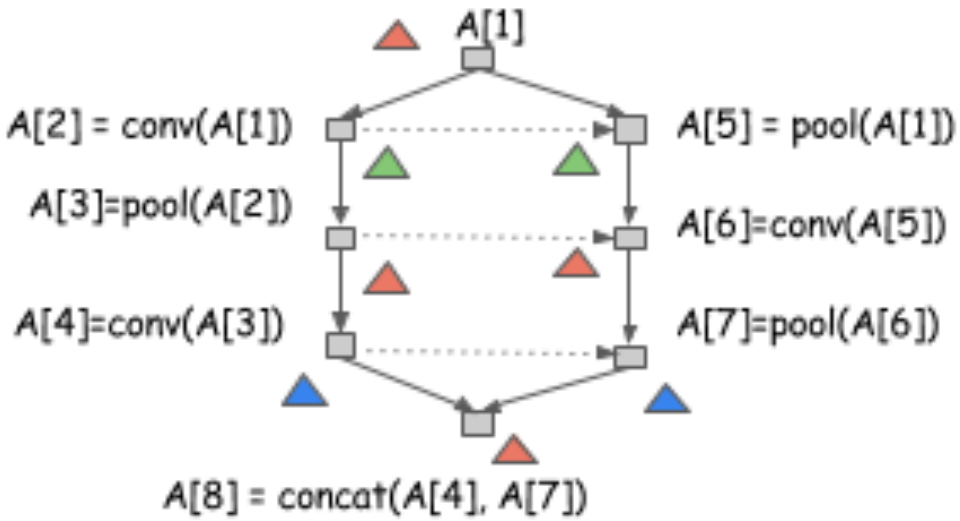
Box of free tags in allocation algorithm.

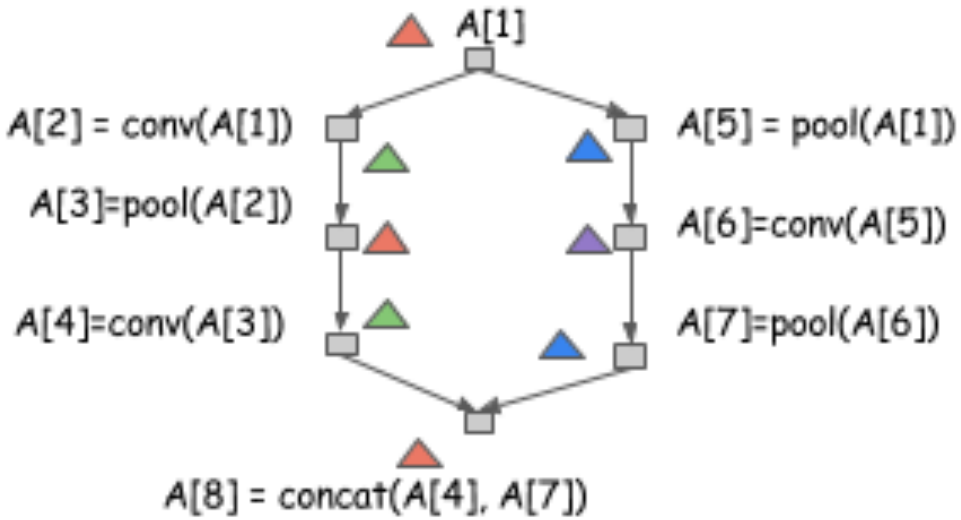
Question?

- 刚才的分配逻辑主要是串行的，遇到并行会怎么样呢？

并行操作上分配

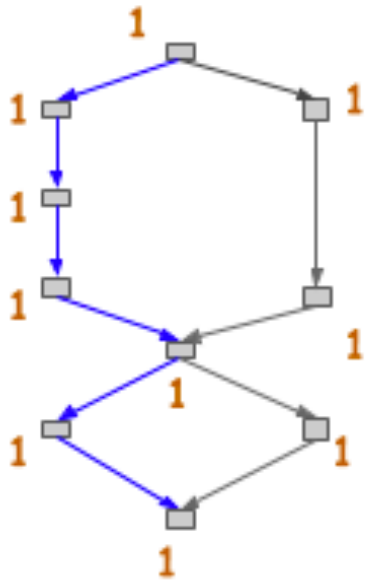
- 以串行方式运行从A[1]到A[8]，那这两种分配方案是都有效的。然而，左侧的分配方案引入更多的依赖，意味着我们不能以并行方式运行A[2]和A[5]的计算，而右边可以运行。



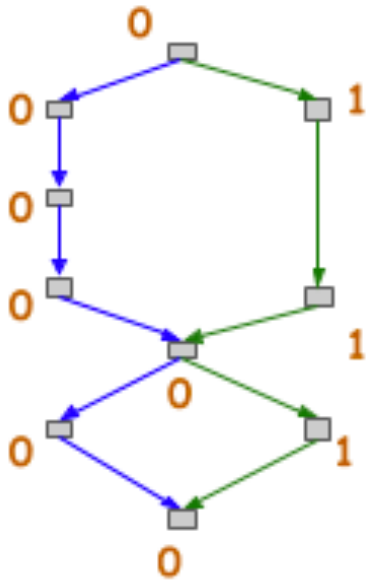


尽量允许更多的并行

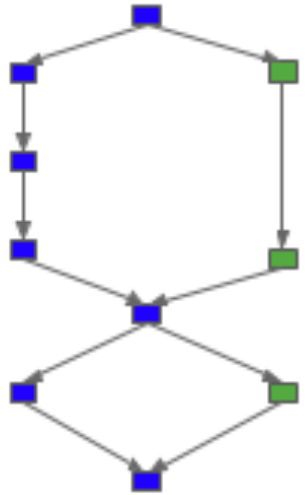
- 尽量鼓励在无法并行节点之间做内存共享，如在分配时通过创建和查询一个父辈关系图，每次尝试找到一个图中最长路径，再进行Inplace和co-share操作。



First the Longest Path



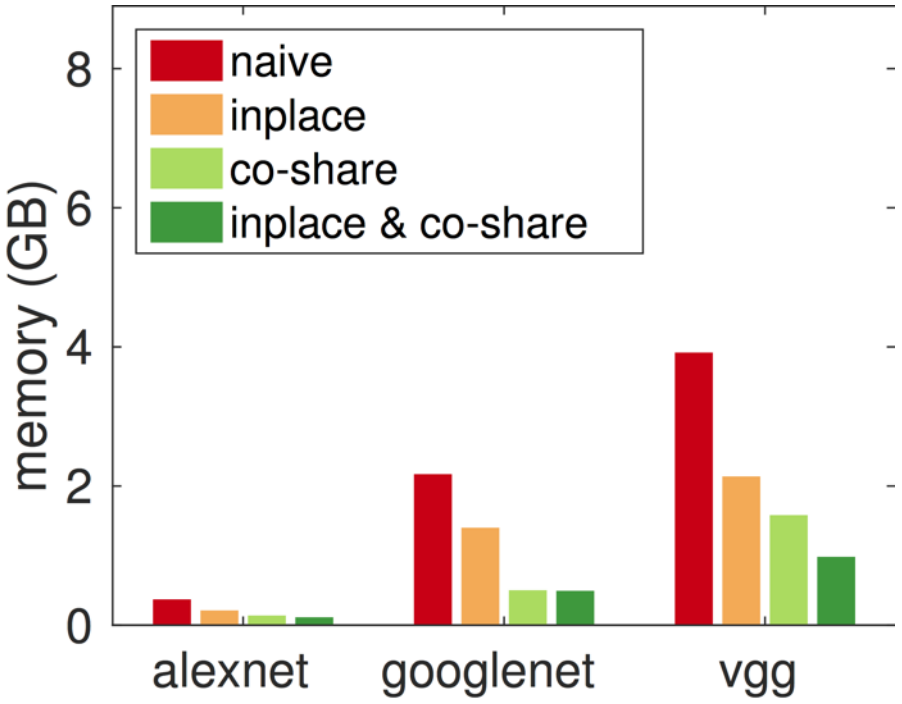
Reset the Reward of visited Node to 0. Find the next longest Path

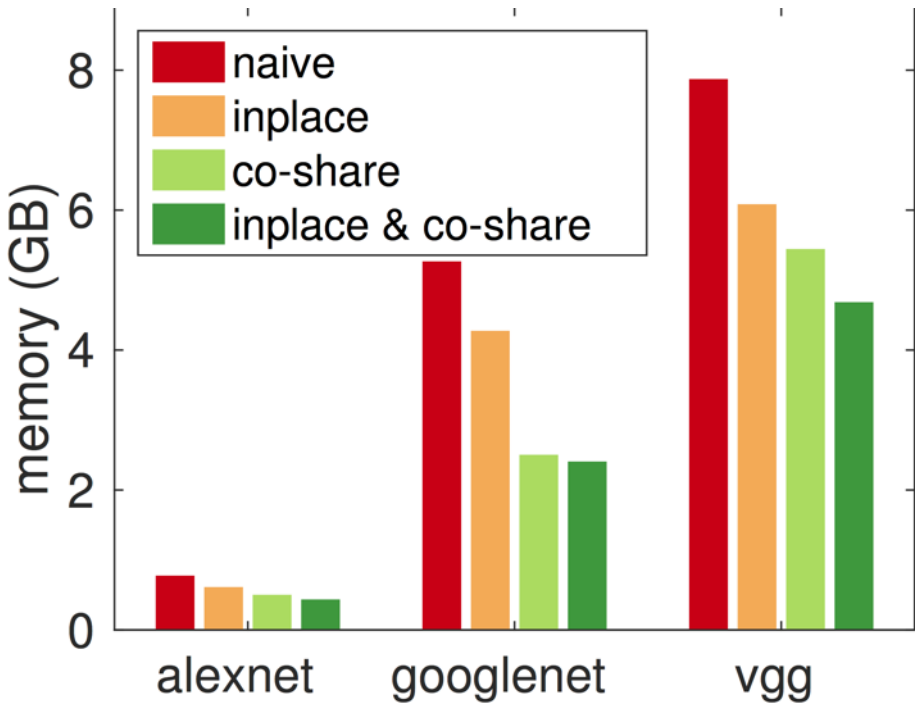


The final node Color

实验结果对比

- Inplace和co-share两者都可以极大的降低内存使用。将两者合起来可以在训练时减少2倍内存使用，在预测时则可以减小4倍内存使用。特别的，即使是最复杂的vggnet，对单张图片进行预测时，只需要16MB额外内存。





Summary

- 使用AI编译器利用 Graph IR 可以巧妙和正确地分配内存。
- 运行深度学习推理的消耗比深度学习训练的内存消耗要少得多。

Reference

1. Chen, Yanbei, Xiatian Zhu, and Shaogang Gong. “Semi-supervised deep learning with memory." *Proceedings of the European conference on computer vision (ECCV)*. 2018.
2. Chen, Tianqi, et al. "Training deep nets with sublinear memory cost." arXiv preprint arXiv:1604.06174 (2016).
3. Optimizing Memory Consumption in Deep Learning. https://mxnet.apache.org/versions/1.9.1/api/architecture/note_memory
4. 机型与性能. https://docs.ucloud.cn/gpu/type.pdf