Qualcomm

MOCCASIN: Efficient Tensor Rematerialization for Neural Networks

Speaker: Burak Bartan

Senior Engineer

Qualcomm Technologies, Inc.

*Qualcomm AI Research**

Email: bbartan@qti.qualcomm.com

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Introduction

- The deployment and training of neural networks on edge computing devices pose many challenges.

- The low memory nature of edge devices is often one of the biggest limiting factors encountered in the deployment of large neural network models.

- Tensor rematerialization or recompute is a way to address high memory requirements for neural network training and inference.

Introduction

- Consider execution of the nodes in the order

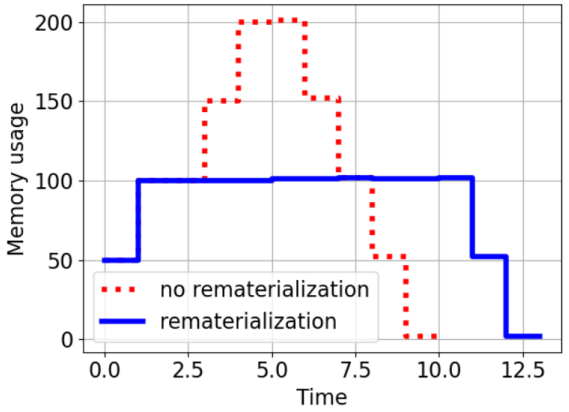
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- We need to keep the output of node 1 in memory until we execute node 8 (similarly for 2&7)
- **Rematerialization:** Discard output of node 1 after node 2 is computed and recompute it later:

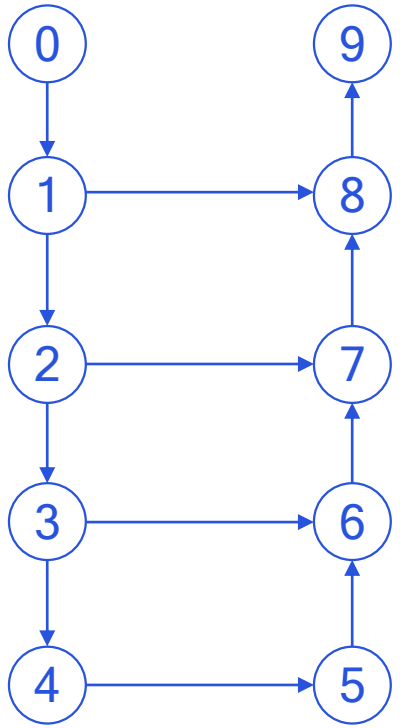
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9

  discard 1 recompute 1

- **To recompute node 1:** We need to recompute 0 or have it available in memory – another decision that needs to be made by the algorithm



Memory usage vs time



10-node computation graph where
- Output sizes are 50 for nodes on the left, and 1 for the ones on the right.
- Duration is 1 for all nodes.

Introduction

- This problem studies the trade-off between memory and compute.
  - By allowing nodes to be rematerialized (i.e. extra compute), it is possible to lower the memory footprint of a sequence.
  - The problem of determining which node outputs to keep in memory and which to discard (and rematerialize later) to reduce peak memory is a combinatorial optimization problem.

MEMORY-CONSTRAINED COMPUTATION GRAPH SEQUENCING WITH REMATERIALIZATION

$$\underset{seq(G)}{\text{minimize}} \quad \text{total execution duration}$$

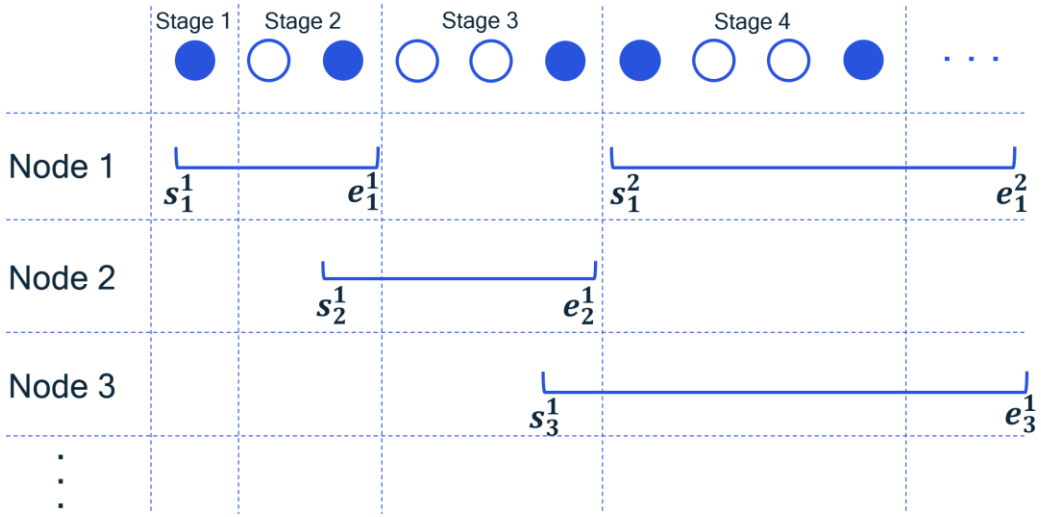
subject to

- $seq(G)$ meets the data dependencies of G
- peak memory footprint of $seq(G)$ $\leq$ local memory capacity M

- We have developed a new optimization problem formulation keeping scalability in mind, which greatly simplifies the problem.
  - Key assumption: Don't rematerialize any given node more than C times.

Proposed solution

- Our solution is an optimization problem formulation where
  - The main building block is the concept of output retention intervals which simplifies the problem formulation greatly
  - Output retention intervals are used to model how long the output of a node is retained in memory
  - We define C retention intervals for each node where each interval represents a rematerialization (i.e. recompute) of that node
  - The starting s_v^i and end e_v^i times of the intervals are modeled as the decision variables of the optimization problem
  - We then define the precedence and memory constraints using these decision variables
  - The resulting problem could be posed as a constraint programming (CP) problem, which we could use any generic CP solver to solve numerically



Proposed solution

$$\underset{s,e,a}{\text{minimize}} \quad \sum_{v,i} w_v a_v^i \qquad (1)$$ ---------▸ Total amount of computation

$$\text{subject to} \quad s_v^i \leq e_v^i,\ \forall v, i \qquad (2)$$ ---------▸ Start of an interval comes before end

$$e_v^i \leq s_v^{i+1},\ \forall v, \forall i < C_v \qquad (3)$$ ---------▸ Ordering among the intervals of a node

$$\sum_{v,i\,:\,s_v^i \leq t \leq e_v^i} m_v a_v^i \leq M,\ \forall t \in \mathcal{D} \qquad (4)$$ ---------▸ Memory use cannot exceed M

$$\forall (u,v) \in E,\ \forall i \in \{i : a_v^i = 1\},\ \exists j \text{ such that}$$

$$a_u^j = 1, s_u^j + 1 \leq s_v^i \leq e_u^j \qquad (5)$$ ---------▸ Precedence constraints

$$s_v^i \neq s_u^j,\ \forall v, u \in \{v, u : v \neq u\},\ \forall i, j \qquad (6)$$ ---------▸ Interval start times are all different

$$a_v^1 = 1,\ \forall v \qquad (7)$$ ---------▸ There is at least one active interval per node

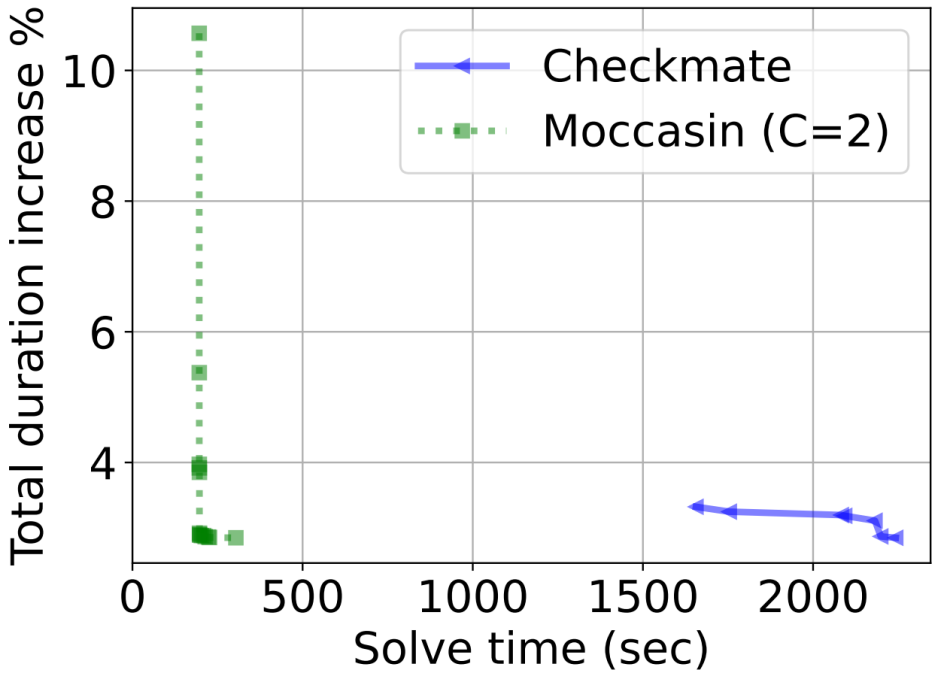
$$s_v^i, e_v^i \in \mathcal{D},\ a_v^i \in \{0,1\},\ \forall v, i\,. \qquad (8)$$ ---------▸ Variable domains

Notes:
- The formulation above does not enforce any ordering on the nodes.
- We consider a fixed topological order to speed up solve.
- Number of intervals does not have to be a global parameter: C_v

Discussion and results

- The concept of retention intervals simplifies the problem formulation greatly
- More concretely, consider the complexity comparison:

- <u>Comparison:</u>
  - Our formulation: $2Cnlog(n)$ Boolean variables
  - Checkmate: $O(n^2 + nm)$ Boolean variables



Real-world graph with $n = 442$ nodes and $m = 1247$ edges

Thank you

Qualcomm

Follow us on:

For more information, visit us at:

qualcomm.com & qualcomm.com/blog