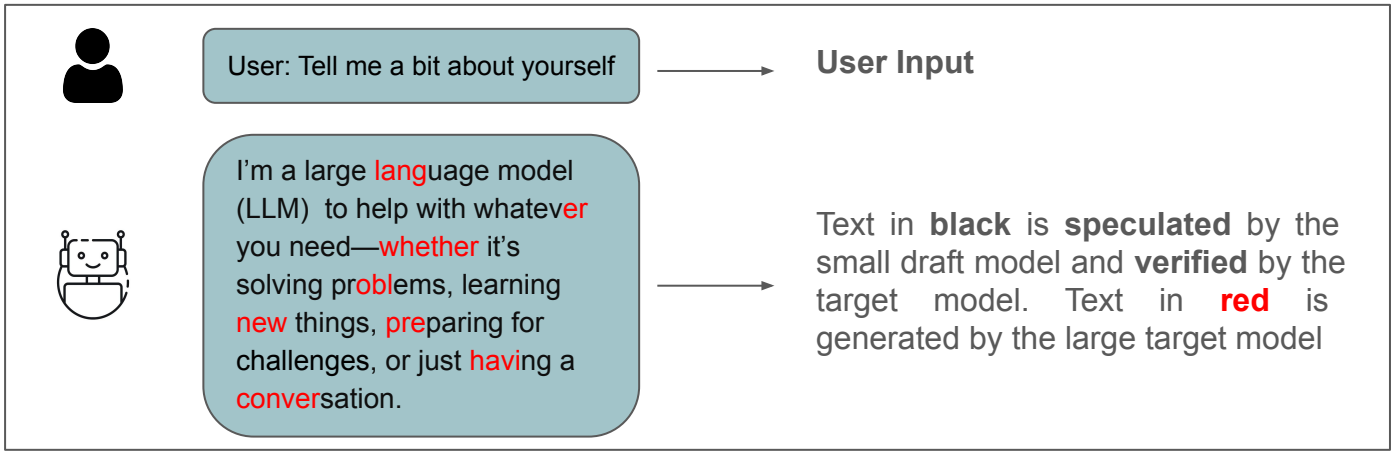
Dynamic Tree-based Speculative Decoding in Multi-GPUs Setup

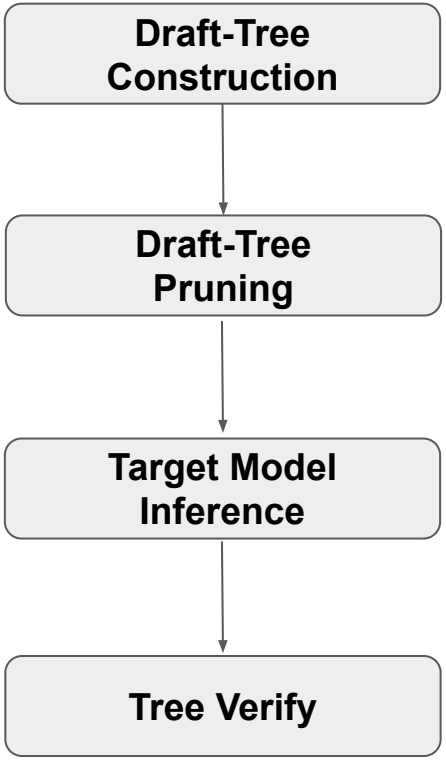
Tengda Wang (tengdaw), Mingxiao Huo (mingxiah)

We implemented **dynamic tree-based speculative decoding** in **multi-GPU** environments using **PyTorch** and **CUDA**

Feature: Parallelize the **draft tree construction**, **tree pruning**, **target model inference** steps
Result: Achieve up to **2.05x** speedup in **4 RTX-A6000 GPUs** compared to **a single GPU**



Overview of Dynamic Tree-based Speculative Decoding



A small **draft** model generates candidate tokens organized in a tree structure. Each tree node represents a distinct token sequence. The tree structure helps increase the accepted length of the tokens generated by the draft model.

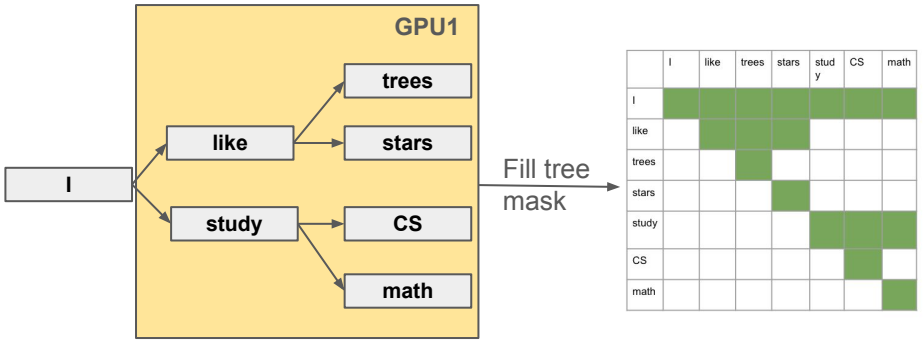
Selectively prune the token tree based on logits of the node and expand only for more likely paths.

The large **target** model run one-step inference on the pruned token tree and generate output logits for all nodes.

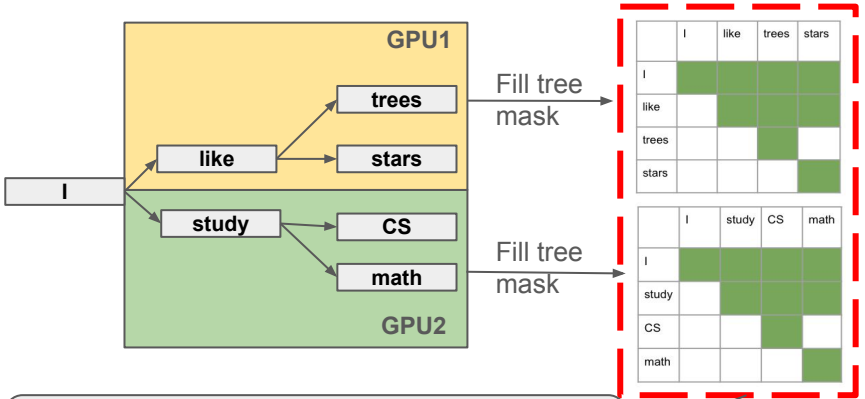
Greedily traverse the token tree and verify the candidate tokens based on node value and logits of target model at the node . This ensures the draft token sequence will be the same as the target token sequence.

Tree-Construction Parallel

Single GPU



Multiple GPUs

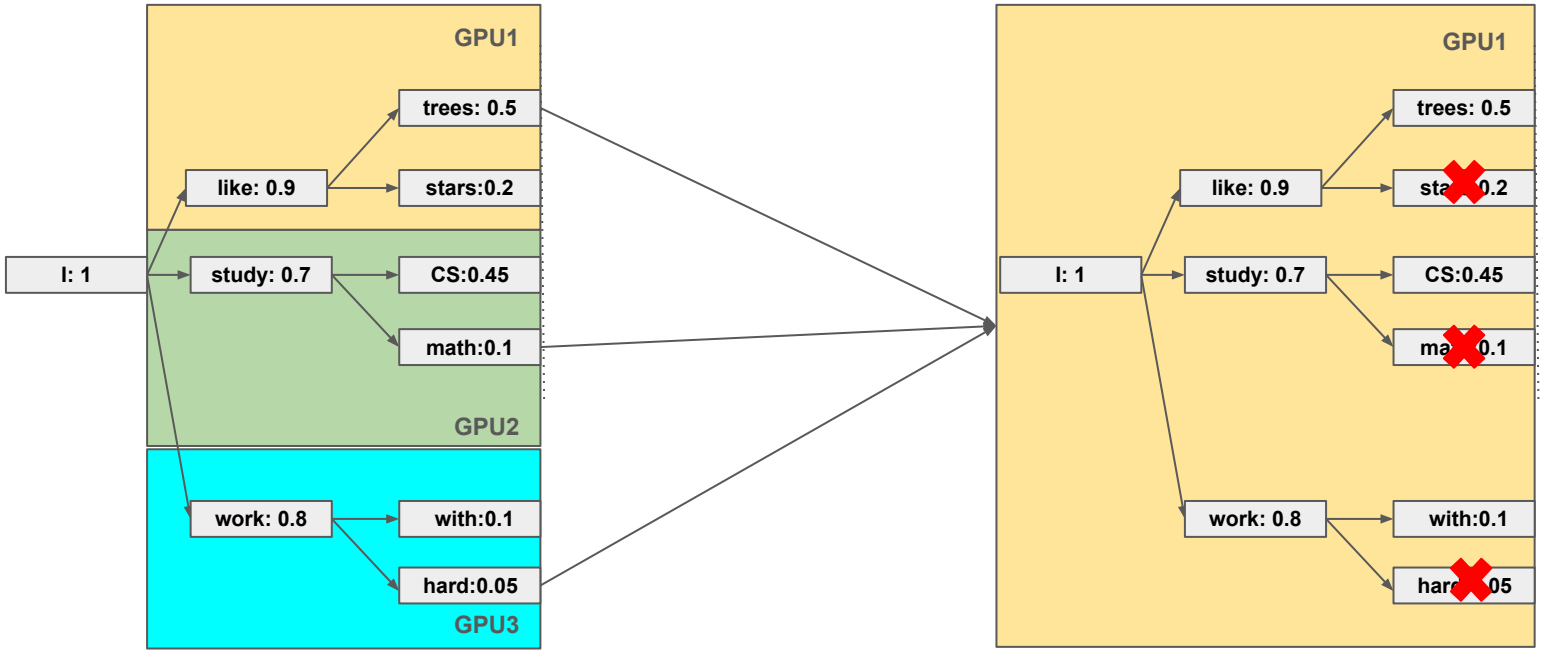


The tree mask construction is supported by **CUDA** utilizing nodes independency

Once the tree is generated, different branches are distributed across GPUs.
The tree masks and node values are filled in **independently** on multiple GPUs using **data parallel**.

Tree-Pruning Parallel

Objective: Select top-K nodes that maximize a value function



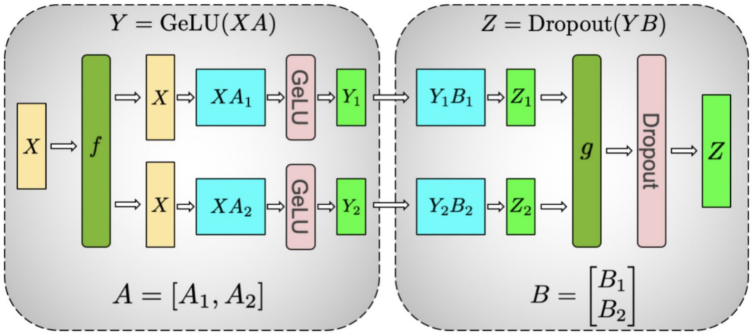
Step 1: Compute value functions for each node independently on each branch

Step 2: Inter-GPU communication

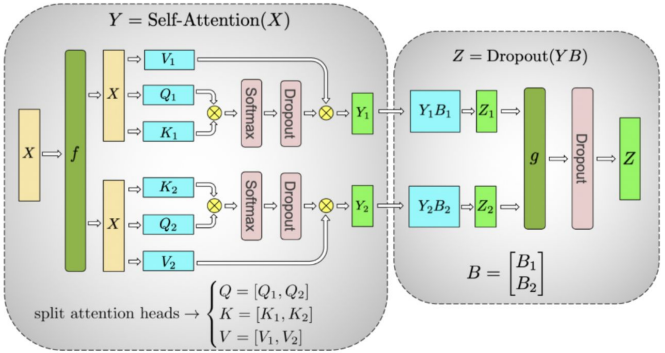
Step 3: Prune the tree on a single GPU

Target Model Inference Parallel

For the large target model, we used **tensor parallel** to distribute the model across multiple GPUs



(a) MLP

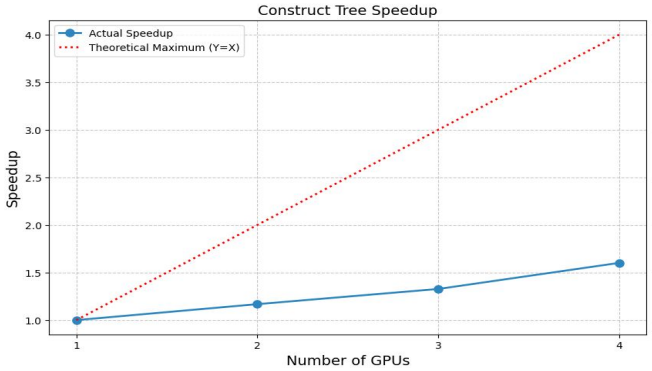


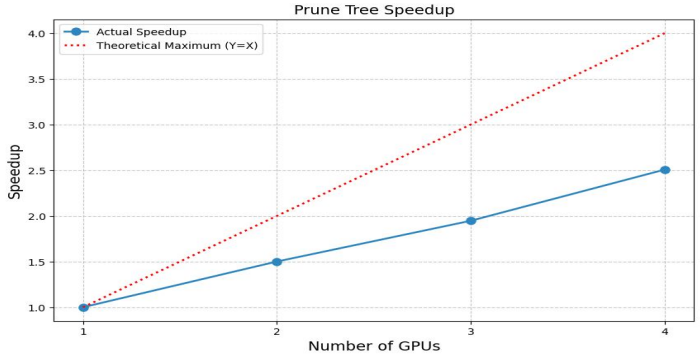
(b) Self-Attention

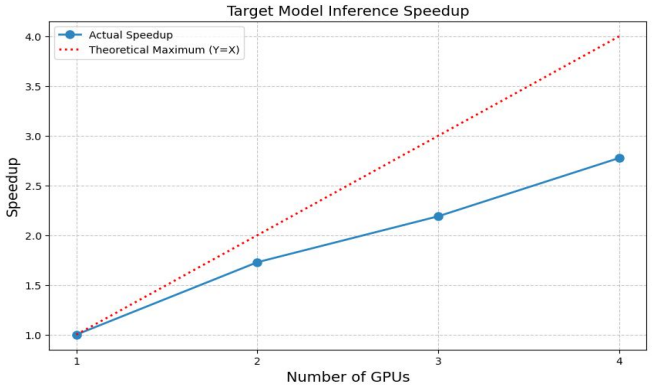
For the **linear** layers, the weight matrices are divided and distributed across gpus.

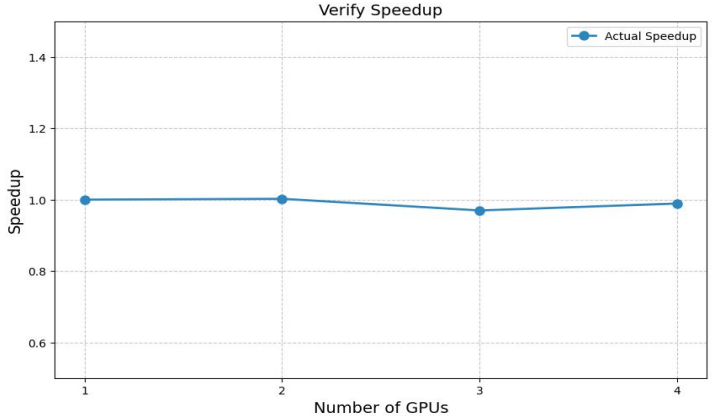
For **multi-head** attention layers, attention heads are distributed across gpus.

Speedup of Different Stages



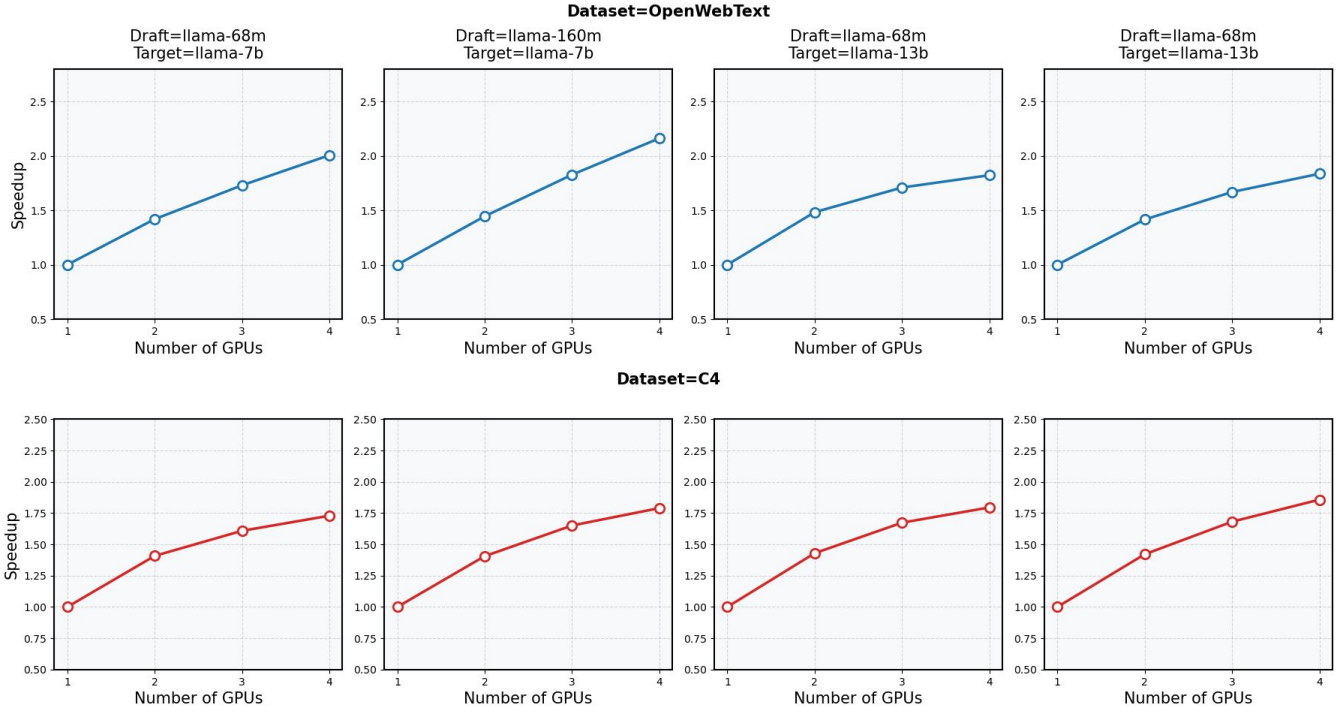






We observe speed up for all stages except the verification part, as it is inherently sequential

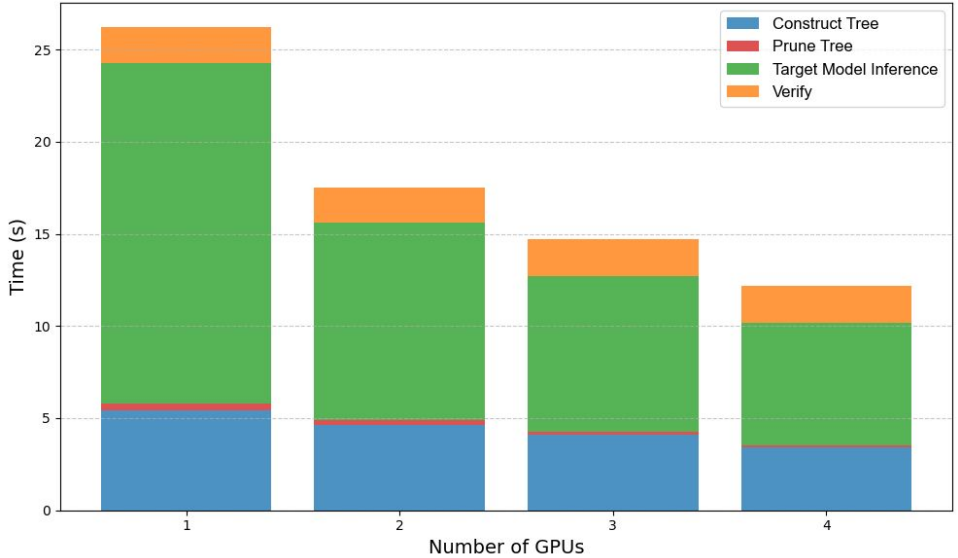
Putting Everything Together



We observe consistent speedup pattern across different **datasets**, **target models**, and **draft models combination** in multi-GPU environments

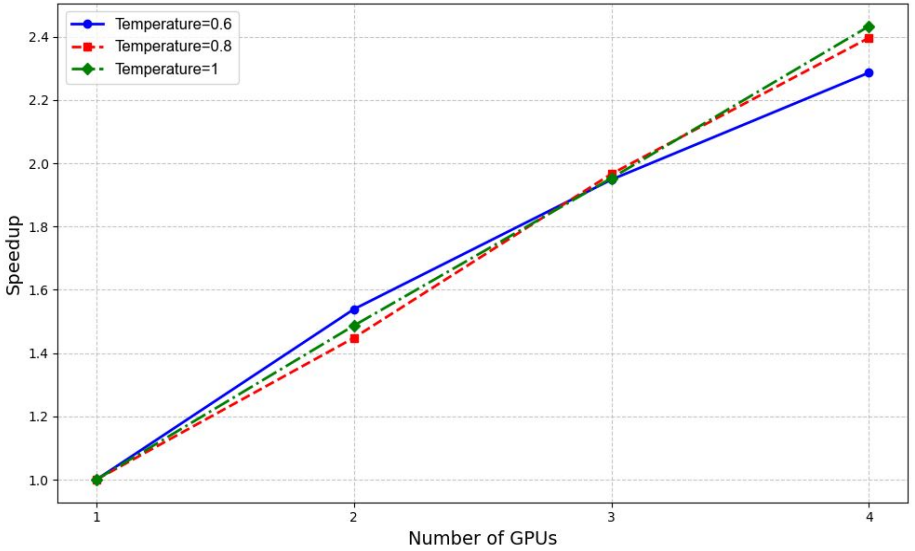
Discussion

Runtime Breakdown



Target model inference is most time consuming.
Verification step is inherently **sequential**.

Sensitivity



Our solution is **robust** to different **temperatures**, which controls the sampling confidence of the model.