Ref: https://sites.google.com/view/medusa-llm
https://ostin.tistory.com/403
https://github.com/FasterDecoding/Medusa
https://www.themoonlight.io/en/review/medusa-simple-llm-inference-acceleration-framework-with-multiple-decoding-heads

**speculative decoding** = employs a streamlined "draft" model to generate a batch of token candidates at each step quickly.

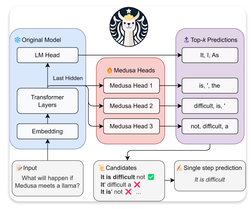
-validated by the original, full-scale language model to identify the most reasonable text continuations
=>draft model, although smaller, should be proficient enough to churn out sequences that the original model will find acceptable.

per-token prices for every major LLM API




**Medusa** is a specific and highly optimized variant of **Speculative Decoding**, a technique designed to dramatically accelerate the text generation process of large language models (LLMs). Traditional speculative decoding typically employs a single, smaller "draft model" to predict a sequence of subsequent tokens, which is then verified by a larger, more accurate "target model."
Medusa takes this idea several steps further.

### Overview of Medusa




### Medusa Structure

**Additional Heads to Existing LLM:**
Medusa Heads are added to the final layer of an already trained Large Language Model (LLM)

When the original head predicts the $t$-th token, the $k$-th Medusa head is trained to predict the $(t+k)$-th token.

Medusa utilizes multiple "heads." These heads are small, lightweight predictors attached in parallel to the final layers of the *target model itself*. Each head is responsible for predicting a subsequent token based on the current context.

**Parallel Prediction and Verification:** Each head simultaneously attempts to predict multiple future tokens. For instance, one head might predict the next token, another head might predict the token *after that*, and so on, with N heads attempting to predict N tokens in parallel.

**Future Token Prediction:**
Each Medusa Head is trained to predict tokens at a pre-defined step into the future, ranging from the very next token up to the $k$-th subsequent token.

**Parallel Draft Generation:**
- During inference, the Medusa Heads concurrently predict multiple future tokens in parallel to generate a draft sequence.

**Verification by Main LLM:**
- The generated draft sequence is passed to the Main LLM all at once for parallel verification.

**Token Confirmation:**
- Tokens determined to be valid are confirmed and outputted simultaneously.

**Tree-Structured Candidate Sequences:** The tokens predicted by these heads form a tree-like structure of various possible token sequences (paths). From the current token, Medusa generates multiple potential future token sequences simultaneously (e.g., $T_1 \rightarrow T_2 \rightarrow T_3, T_1' \rightarrow T_2'' \rightarrow T_3'$, etc.).

**Batch Verification by Target Model:** The target model then performs a single forward pass to batch-verify all these possible candidate sequences. It calculates the probabilities for each token within each candidate sequence and identifies the most probable, correct sequence.

**Accepting Longer Sequences at Once:** Medusa significantly increases the probability of "accepting" a much longer sequence of tokens in a single verification step. While standard speculative decoding verifies tokens sequentially or in limited-length sequences, Medusa explores more parallelism through its tree structure, allowing it to potentially skip many tokens at once.

### Why we should use Medusa?

- **Maximized Speedup:** By enabling significantly more parallel prediction and verification than traditional speculative decoding, Medusa can achieve much greater accelerations in text generation speed. It is particularly optimized for parallel processing hardware like GPUs.

- **Preservation of Quality:** Like all forms of speculative decoding, Medusa is a **lossless** optimization. The final generated text is **mathematically identical** to what the target model would have produced on its own. Even if the draft heads predict incorrectly, the target model corrects these errors and guarantees the correct output.

- **No Separate Draft Model Needed:** Unlike traditional speculative decoding, which often requires training or acquiring a separate, smaller draft model, Medusa integrates small heads directly into the target model's final layers. This can simplify model management and deployment.

**Medusa Heads :** Instead of relying on a separate smaller "draft model" as in speculative decoding, MEDUSA adds K extra decoding heads to the last hidden state of the original LLM. The k-th head predicts the token at position t+k+1, where t is the current position.

**Implementation**: Each head consists of a single-layer feed-forward network with a residual connection:

$p(k)t = softmax(W(k)2 \cdot (SILU(W(k)1 \cdot ht) + ht))$

- $ht$: last hidden state at position $t$.
- $W(k)1$: a $d \times d$ matrix, initialized to zero.
- $W(k)2$: a $d \times V$ matrix, initialized with the language model head's weights. ($d$ is the hidden dimension, and $V$ is the vocabulary size).
- $SILU$: the activation function used.

**Benefits:**

Parameter-efficient (only adds a small number of parameters).
Easily integrated into existing LLM systems.
Avoids distribution shift issues that can arise with separate draft models, since the MEDUSA heads are either trained alongside, or simply added to the backbone model.

Refers to a form of attention mechanism designed primarily to process **tree-structured data**. Standard Self-Attention in typical Transformer models is optimized for processing sequential data; that is, each token interacts with all other tokens within the sequence to compute attention weights.

**Core Idea of Tree Attention =** making the attention mechanism **leverage the tree structure information of the data**

1 **Encoding Tree Structure:** The tree structure of the input data is explicitly incorporated into the attention calculation. This is typically done in ways such as:

- **Tree-based Attention Masking:** When computing attention weights, a mask is applied so that only nodes with specific relationships on the tree (e.g., parent-child, ancestor-descendant, within the same subtree) can attend to each other. Alternatively, attention weights can be adjusted based on the path length on the tree.
- **Adding Tree-related Embeddings:** To each node's (token's) embedding, additional embeddings representing its structural position in the tree (e.g., depth, parent node information, subtree information) are added.
- **Tree-informed Query (Q), Key (K), Value (V) Generation:** The method of generating Q, K, and V vectors is modified to reflect the tree structure.

**2. Capturing Hierarchical Information:** This allows the model to learn and utilize the data's **hierarchical and structural relationships** more effectively, going beyond merely linear positional relationships.

**Tree Attention:** MEDUSA heads generate multiple candidate continuations, and instead of processing them sequentially, the method employs a tree-based attention mechanism. This allows for parallel verification of these candidate tokens