CALID: Collaborative Accelerate LLM Inference with Draft Model with Filter Decoding

Yifan Hua [1] **Shengze Wang** [1] **Xiaoxue Zhang** [2] **Daniel Zhu** [3] **Arthur Cheong** [4]
Karan Mohindroo [5] **Allan Dewey** [1] **Chen Qian** [1]
[1]UC Santa Cruz [2]University of Nevada, Reno [3]The Harker school
[4]Mountain View High School [5]Pierrepont School
{yhua294,shengze,acdewey, cqian12}@ucsc.edu xiaoxuez@unr.edu
{danielfangzhu,karankmohindroo,arthurcheongmvhs}@gmail.com

1 Background and Motivation

Large language models (LLMs), such as autoregressive Transformer models Vaswani et al. (2017), including GPT Brown et al. (2020) and LLaMA Touvron et al. (2023), drive a wide array of applications. These models, which typically contain billions of parameters, are the backbone of many LLM-based services. However, processing a single LLM request can cost up to ten times more than a traditional keyword search Dastin and Nellis (2023). This significant cost emphasizes the urgent need to reduce operational costs in cloud environments to meet the growing demand for LLM services.

To mitigate the underutilization of GPU parallelism, researchers have proposed Speculative Decoding Leviathan et al. (2023); Chen et al. (2023), which uses smaller models to produce initial drafts. The main LLM refines these drafts into a batch of inputs for a single request, as shown in Fig. 2b. However, speculative decoding can decrease the throughput because 1) the parallelism is already utilized and 2) it generates numerous unnecessary drafts for the large target model to evaluate, causing a waste of computational resources. To eliminate such efficiency, we propose CALID which uses a confidence score based filter decoding mechanism to avoid the unnecessary inferences of the large model while utilizing the efficient small language model (SLM) to generate the results.

2 System Design

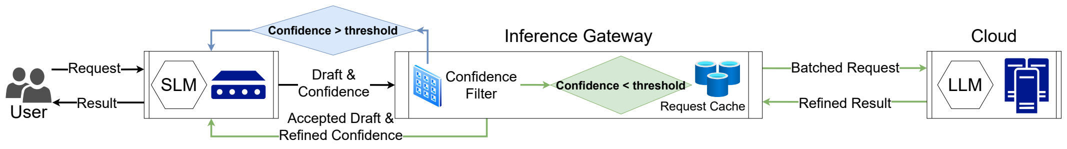


Figure 1: CALID System Design

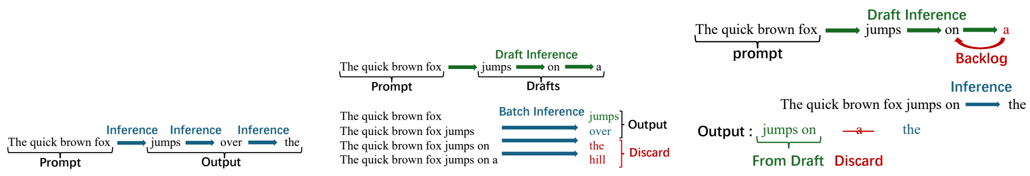
We introduce the architecture of CALID in this section, and illustrate the system design in Fig. 1.

2.1 Negative Log-Likelihood (NLL) based confidence score

The confidence score system is based on a negative log-likelihood (NLL) based metric, which provides a significant indication of the output quality produced by an SLM. To compute the NLL-based confidence score, we establish a probability distribution for potential subsequent tokens as $t_1, t_2, ..., t_{maxk}$, ranked in descending likelihood in the distribution of all tokens, P_i. The Confidence Score is calculated by the most likely token as Equation 1, which serves as an estimator for the output quality of the draft model.

$$Confidence(P_i) = NLL(t_1) \tag{1}$$

The probability associated with the most likely candidates intuitively serves as a measure of 'confidence' in the predictions made by the language model. Consider a programming assistant for code completion as an example. To complete a for-loop statement '*for (int i = 0; i < n;*', a preliminary draft model might predict the tokens '*i*' and '++' with high likelihood. In these scenarios, the confidence score C_1 should be high, offering a reliable estimation of the output quality from the draft model.



(a) Vanilla Decoding **(b)** Speculative Decoding **(c)** Filter Decoding

Figure 2: In the decoding process, the blue arrows represent inferences with draft models; the green arrows are inferences with a draft model.

2.2 Filter Decoding

Although large transformer models deliver high-quality outputs and are pivotal in numerous AI-driven applications, the efficiency of autoregressive generation presents a significant limitation. The inherent attention mechanism in transformers requires that generating a sequence of T tokens needs T sequential autoregressive evaluations by the model.

In the traditional decoding process in Fig. 2a, the model first evaluates the prompt as the initial input. Then, a token is sampled from the distribution generated by the model and appended to the preceding input to form the new input.

Speculative decoding, although not efficient, tolerates the variability in the difficulty of inference steps, where some tokens are "harder" and others are "easier". Based on the insights of speculative decoding, we further develop **Filter Sampling**. As shown in Alg. 2, M_p denotes the accurate LLM, M_e denotes the efficient SLM, $input_{1..k}$ denotes the user request prompts, T denotes the output length, and $Threshold_c$ denotes the adaptive threshold to balance output quality and throughput. We also provide the vanilla sampling method of Auto-Regressive Sampling in Alg. 1 for comparison.

Algorithm 1 Auto-Regressive Sampling

```
Input: $M(.|.)$, $input$, $T$
Initialize $n \leftarrow t$
**for** $i = 1$ **to** $T$ **do**
    $P_i \leftarrow M_p(x|input + \{x_1, ..., x_{i-1}\})$
    Sample $x_i \sim P_i$
**end for**
**return** $x_1, ..., x_T$
```

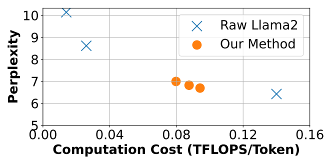


Figure 3

Algorithm 2 Filter Sampling

```
Input: $M_p(.|.)$, $M_e(.|.)$, $input_{1..k}$, $T$, $Threshold_c$
Initialize $n \leftarrow t$
**for** $i = 1$ **to** $T$ **do**
    $Q_i \leftarrow M_e(x|input + \{x_1, ..., x_{i-1}\})$
    $C_i \leftarrow Confidence(Q_i)$
    **if** $C_i > Threshold_c$ **then**
        Sample $x_i \sim Q_i$
    **else**
        $P_i \leftarrow M_p(x|input + \{x_1, ..., x_{i-1}\})$
        Sample $x_i \sim P_i$
    **end if**
**end for**
**return** $x_1, ..., x_T$
```

$Confidence(\cdot)$ function processes the distribution from the SLM to determine the confidence level of its outputs. Outputs from the efficiency model that achieve high confidence scores are deemed high-quality and accepted as final results. Lower-quality outputs are substituted with those from the performance model, optimizing computational resource usage. This methodology ensures the efficient allocation of computational resources, enhancing system's overall performance.

The inference gateway serves as a pivotal component within the system's architecture, functioning as a proxy between clients that utilize SLMs and LLM clusters equipped with high-performance GPUs.

In the standard operation, the gateway aggregates user requests into batches and directs them to the inference clusters. In CALID, the inference gateway initially caches the input from the users with their confidence score in the Request Cache. Then, it executes batch Filter decoding by first calculating the $threshold_c$. In our implementation, $threshold_c$ is set to a constant value for simplicity. During filter decoding, the Inference Gateway selectively batches requests with lower confidence scores, sends them to the LLM Inference backend, and returns an early acceptance of drafts of the requests with high confidence scores.

The preliminary evaluation in Fig. 3 demonstrates that substituting some results with those generated by SLM has minimal impact on the overall quality of the output. Blue crosses represent the performance of Llama2 (7B, 13B 70B from left to right). Orange dots represent our work with different $Threshold_c$. The Inference gateway filters out 47.80%, 41.6% and 36.4% of requests to the Llama2 7b and the rest to Llama2 70b from left to right.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318* (2023).

Jeffrey Dastin and Stephen Nellis. 2023. Focus: For Tech Giants, ai like bing and Bard poses billion-dollar search problem | reuters. `https://www.reuters.com/technology/tech-giants-ai-like-bing-bard-poses-billion-dollar-search-problem-2023-02-22/`

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*. PMLR, 19274–19286.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

3 Acknowledgements

Y. Hua, S. Wang, A. Dewey, and C. Qian were partially supported by NSF Grant 1932447, 2114113, and 2322919.