BTDefense: A Priority-Aware Adaptive Strategy Against the Block Table Overflow Attack in vLLM*

Yuyan Zhao, Menghao Zhang, Tianyu Wo, Renyu Yang, Chunming Hu
Beihang University

1 PROBLEM STATEMENT

vLLM [1] is a large language model (LLM) inference service system that achieves near-zero waste in KV cache memory while enabling flexible sharing of KV cache both within and across requests to further reduce memory consumption [2]. Inspired by paged memory management techniques used in operating systems, vLLM adopts the PagedAttention algorithm and utilizes a block module for memory allocation and management. This approach dynamically allocates KV cache GPU memory for requests, improving memory efficiency and supporting large-scale language model inference operations.

However, large models typically have millions of or even billions of parameters, they require a substantial amount of costly memory resources. In memory-constrained environments, the block table space in most large model inference service systems is relatively limited, and it often proves inadequate when handling a high volume of inference requests. When a large number of requests arrive at the vLLM server, vLLM employs a First-Come-First-Serve (FCFS) strategy, prioritizing the processing of earlier requests. As the number of requests and their outputs increases, vLLM may exhaust GPU memory to store the newly generated KV cache, leading to insufficient GPU memory and ultimately preventing the processing of new requests.

To handle new requests, vLLM temporarily suspends the execution of certain tasks, adopting an all-or-nothing strategy to release all KV cache blocks associated with preempted requests. These tasks are resumed once sufficient GPU resources become available. Although this mechanism helps manage limited block resources and prevents overflow during resource contention, it may lead to the unintended eviction of frequently accessed and important block sequences. Attackers can exploit this mechanism by launching malicious attacks, continuously sending a large number of high-frequency requests to occupy the majority of block resources. By monopolizing available blocks, they could cause *block table overflow*, preventing legitimate users or requests from accessing the necessary memory resources, resulting in a denial of service for vLLM inference or degraded inference performance.

*Supported by National Science Foundation of China (No. 62402025) and the Fundamental Research Funds for the Central Universities. Menghao Zhang is the corresponding author.

2 COUNTERMEASURE ANALYSIS

As all new requests compete for the limited block table resources, subsequent requests may inevitably evict previously allocated memory blocks without discrimination. This issue becomes particularly problematic when malicious users continuously issue a high volume of requests, potentially filling up the entire block table and leading to memory block starvation for other users, who may be unable to access the resources they need. A seemingly viable preliminary solution is to distinguish between suspected and benign requests and to discard the former. However, this approach may not be sufficiently reliable and is susceptible to risks such as false positives and false negatives. As a result, this may result in unavoidable harm to legitimate demands, while permitting some attack requests to bypass protection with ease.

To cope with above problem, we propose BTDefense, a **behavior-based priority adaptive perception** strategy, to address this block table overflow attack. Our basic idea is to distinguish different requests with different priorities and to implement soft isolation for users based on different evaluations. This evaluation is formalized into a score that favors benign request characteristics while disadvantaging malicious ones. Based on these evaluation scores, we dynamically assign different priorities to user requests. Requests from benign users are assigned higher priority, whereas those from malicious users are assigned lower priority. When the block table is full, blocks with the lowest priority are evicted to make space for new requests. Experiments and evaluations demonstrate that BTDefense safeguard the requests from malicious users and effectively mitigates block table overflow attacks in vLLM.

3 BTDEFENSE DESIGN

The architecture of BTDefense is shown in Figure 1. During the design of BTDefense, we encountered three major challenges. 1) how to effectively collect user request behavior. 2) how to establish evaluation criteria to distinguish the requests. 3) how to implement the strategy under limited resource conditions. To address these challenges, we designed three modules, as shown in Figure 1, each corresponding to a solution for the respective challenge.

For the first challenge, we design the *Request Feature Collection* module that gathers user request characteristics from the basic services provided by the vLLM system. Normal

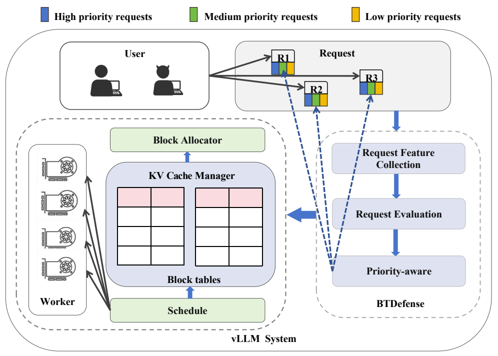

Figure 1: Architecture Overview

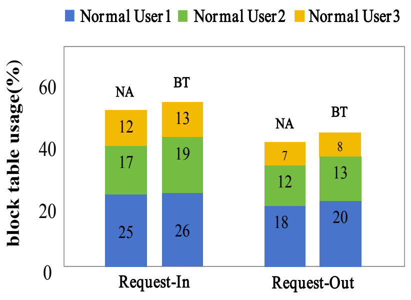

Figure 2: no attack scenarios

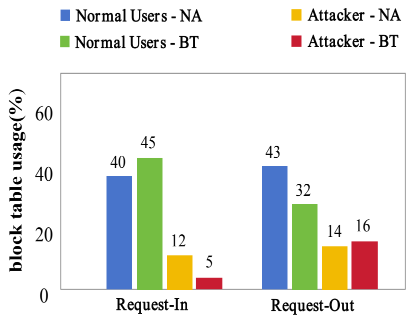

Figure 3: attack scenarios

requests and attack requests typically exhibit differences in aspects such as frequency, resource consumption patterns, geographic location, and request content, with attack behavior often showing obvious anomalies. To maximize system resource consumption, attackers commonly send high-frequency, low-latency requests. By identifying and monitoring request origins (e.g., IP addresses, geographic location), request times, request content, and resource usage characteristics, we can effectively distinguish between benign and malicious requests. When an attacker generates a large amount of requests in a short time, with request structures that deviate from the norm, this behavior can be identified as potentially malicious.

For the second challenge, we introduced a request evaluation module and developed an assessment framework to distinguish between various user behaviors. Based on the collected request feature information, a comprehensive evaluation score is calculated for each request. The importance of each feature is determined through weighted scoring, with the score for different user requests denoted as $S_{request}$, where $S_{request} = \sum_{i=1}^{n} w_i \cdot F_i$, F_i represents the score of the different features, such as request frequency, resource consumption, etc., and w_i denotes the weight of the different features. Considering the user's behavioral history, the weights of each feature are dynamically updated using Exponentially weighted moving average (EWMA) [3]: $w_i = (1 - \alpha)w_i + \alpha \cdot w_i^{\text{new}}$, w_i represents the current weight, w_i^{new} denotes the new weight value, and α is a smoothing factor that lies between 0 and 1.

As for the third challenge, a priority-aware module was introduced to map the request evaluation scores to three priorities (high, medium, and low). These priorities correspond to three request types (benign, suspicious and malicious) respectively. Based on historical data and the distribution of request features, initial thresholds for high priority T_{high} and low priority T_{low} are set. Requests with evaluation scores above the high-priority threshold are classified as benign and assigned a high priority, while requests with scores between T_{low} and T_{high} are considered suspicious and given medium priority. Requests with scores below the low-priority threshold are deemed malicious and assigned a low priority. During the inference process in a vLLM system, block table resources are allocated based on the priority of the requests. High priority requests receive more resources, while medium and low priority requests are limited or delayed. When block table resources are exhausted, low-priority blocks are removed first to make space for new requests, reducing the impact of block table overflow attacks and preventing malicious users from consuming excessive system resources. To adapt to evolving request patterns and attack behaviors, the system dynamically adjusts thresholds based on the load conditions and the nature of incoming requests.

4 EVALUATION AND FUTURE WORK

We implement BTDefense based on the open-sourced vLLM project and perform experiments using LLaMA. We uniformly set the block table size to 16 and simulate user memory usage under different priority requests using three sets of user requests. In Figure 2 and 3, *NA* denotes the original strategy and *BT* refers to our priority-aware strategy.

In the no-attack scenarios, as shown in Figure 2, the system can allocate block resources effectively for normal user requests, demonstrating that our strategy does not introduce any negative impact on benign requests when there is no attack. In the attack scenarios depicted in Figure 3, when a user becomes the attacker and sends a large volume of malicious requests, the attacker's requests will occupy most of the block table space without our strategy, resulting in block table overflow.

In our future work, we plan to explore additional user and request features, refine our evaluation criteria for greater accuracy, provide further recommendations on strategy parameters, conduct more experiments to assess the load on the block table, and investigate more complex system states.

REFERENCES

[1] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica.

2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[2] Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. 2024. Power-aware Deep Learning Model Serving with {μ-Serve}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 75–93.

[3] J. Yu, S. B. Kim, J. Bai, et al. 2020. Comparative study on exponentially weighted moving average approaches for the self-starting forecasting. *Applied Sciences* 10, 20, 7351.