SPADA: Secure, Performant, and Distributed LLM Inference

kexin Chu , Jianchang Su , and Wei Zhang*

School of Computing, University of Connecticut
{kexin.chu, jianchang.su, wei.13.zhang}@uconn.edu

***Abstract*—Large Language Models (LLMs) have achieved remarkable success across a range of applications, from code generation to conversational AI. As LLMs grow in size and capability, distributed inference across multiple computing nodes becomes necessary to meet resource demands and performance goals. However, this shift introduces critical security challenges, particularly in the handling of sensitive user inputs and intermediate model states like key-value (KV) caches. In this paper, we present SPADA—a Secure, Performant, and Distributed Architecture for LLM inference that addresses the core challenges of secure execution, inter-node trust, and efficient communication in distributed environments. SPADA integrates trusted execution environments (TEEs), a decentralized trust establishment protocol, and a lightweight, encrypted communication pipeline. It also introduces a secure and efficient mechanism for transmitting distributed KV cache data. Our design ensures that distributed inference pipelines maintain strong privacy guarantees without sacrificing throughput or latency, offering a practical foundation for secure LLM deployment at scale.**

I. Introduction

Large Language Models (LLMs) have emerged as transformative technologies powering applications like open-domain question answering, code generation, document summarization, and multi-turn conversational AI [3], [7], [20]. As these models grow in size and complexity—spanning hundreds of billions of parameters—the computational and memory overhead for inference has increased significantly [8], [9], [12], [23], [36]. To support real-time applications, substantial effort has gone into performance optimization. Techniques like mixed-precision inference, model quantization, speculative decoding, and key-value (KV) cache reuse are crucial to scaling LLM inference. The adoption of high-performance hardware accelerators like GPUs, TPUs, and custom inference chips has improved throughput and latency. However, as LLM-based applications demand longer contexts and greater interactivity, these optimizations are nearing their limits, especially given the finite memory and bandwidth of a single compute node. Distributed LLM inference—partitioning the model and its state across multiple nodes—has become necessary to scale workloads efficiently [1], [15], [17], [38], [41].

While distributed inference offers performance benefits, it introduces critical security and privacy challenges [6], [19], [29], [33]. Decentralizing inference creates new attack surfaces: untrusted nodes may tamper with intermediate results, infer sensitive input prompts via timing or access patterns, or impersonate other nodes [33], [35], [45]. KV caches, storing intermediate attention keys and values for multi-turn interactions [16], [31], [39], [44], often contain sensitive information [14], [18], [21], [25], [43]. Transmitting these caches without protection poses significant risks. Research has primarily focused on performance and scalability [2], [5], [10], [11], [13], [22], [26], [30], [32], [34], [42], leaving security gaps. This issue is amplified in multi-tenant or cloud-hosted environments, where infrastructure is shared among untrusted parties, and sensitive data may be exposed or leaked [24].

Secure distributed inference requires addressing several design challenges. Participating nodes must verify trustworthiness before exchanging sensitive model states or user data. Without guarantees, compromised nodes could disrupt execution or exfiltrate information. Strong intra-node isolation is essential to preserve model integrity and confidentiality, particularly with co-located workloads or privileged software. Inter-node communication must be encrypted and authenticated to prevent eavesdropping, with minimal overhead [37]. Meeting these goals without degrading performance, especially for latency-sensitive applications, remains a complex problem.

To address these challenges, we present **SPADA** — a **S**ecure, **P**erformant, and **D**istributed **A**rchitecture for LLM inference. SPADA ensures end-to-end security for distributed inference without compromising throughput or latency. It leverages Trusted Execution Environments (TEEs), such as Intel SGX and AMD SEV, to isolate the inference runtime, model parameters, and KV cache within each node, ensuring privileged software (e.g., hypervisors or OS kernels) cannot access or tamper with sensitive state. SPADA introduces the **Decentralized Trust Establishment Protocol (DTEP)** — a lightweight protocol enabling mutually authenticated trust across distributed TEEs without a centralized authority. This design supports scalable deployments where nodes can join or leave the system dynamically.

SPADA also incorporates a high-performance secure communication layer optimized for cross-node KV cache exchange. This layer ensures confidentiality and integrity with authenticated encryption while minimizing serialization and transmission overhead. To enhance performance, SPADA supports parallelized cache fetches and pipelined token execution, ensuring security mechanisms do not become a bottleneck.

Despite these strengths, SPADA introduces new engineering

* Corresponding Author.

challenges. TEEs are memory-constrained and may introduce syscall or paging overheads that affect inference speed. Designing a decentralized trust protocol robust to node churn, failure, or compromise requires careful engineering and secure key management. Synchronizing KV caches across enclaves requires an efficient strategy to maintain consistency and avoid latency spikes or deadlocks.

Our contributions are as follows:

- We identify key privacy and security challenges in distributed LLM inference, including threats from KV cache sharing, inter-node communication, and untrusted infrastructure.
- We propose SPADA, a new architecture that integrates TEE-based isolation, decentralized trust establishment, and secure inter-node communication to ensure both security and performance.
- We design and implement DTEP, a novel trust protocol that supports dynamic, scalable trust relationships across secure inference nodes.
- We develop a secure KV cache transport pipeline that introduces minimal overhead while providing strong privacy guarantees.

II. Background and Motivation

A. LLM Inference

Large Language Models (LLMs) based on the Transformer architecture use dot-product attention to compute Query (Q), Key (K), and Value (V) embeddings that capture token relationships. Inference is divided into two phases: prefill and decoding. In the prefill phase, the input prompt is tokenized and processed in parallel to compute and store K/V embeddings as the KV cache. In the decoding phase, tokens are generated autoregressively; each new token's Q attends to cached K/Vs, avoiding recomputation. Without KV caching, inference would have quadratic time complexity, making long prompts and multi-turn interactions impractical. KV reuse is thus essential for low-latency, high-throughput inference in real-time applications.

B. TEE

Trusted Execution Environments (TEEs) are secure areas within modern processors that provide isolated execution environments to protect the confidentiality and integrity of data and code. TEEs operate by creating hardware-enforced boundaries that prevent unauthorized access from other software on the same system, including the operating system, hypervisor, or even physical attackers with root privileges. Technologies such as Intel Software Guard Extensions (SGX) and AMD Secure Encrypted Virtualization (SEV) enables sensitive computations to run securely within these enclaves, shielding them from a wide range of threats. TEEs are particularly valuable in cloud and edge computing settings, where resources are shared across multiple tenants and trust assumptions are limited. By offering hardware-based isolation, TEEs enable secure processing of sensitive data without requiring trust in the underlying system software or infrastructure.

C. Motivation and Challenges

TABLE I: Personal Information Counts in C4 and Pile [4].

Personal Information Type	C4	Pile
User Name	1,444,683,066	3,273,163,949
Phone Number	19,592,273	23,191,595
Email Number	9,056,833	13,336,793
US Bank Number	7,139,838	69,763,678
Credit Card Number	61,405	741,815
US SSN	2,352,339	12,541,022
IP Address	1,890,090	14,975,663
Total	1,484,780,621	3,407,722,116

As shown in Table I, the analysis of the C4 and Pile corpora reveals an abundance of personally identifiable tokens [27]——over 1.44 billion user names, nearly 20 million phone numbers, 9 million email addresses, 7 million U.S. bank numbers, and millions more credit-card numbers, SSNs and IP addresses. Such pervasive distribution of sensitive data imposes stringent security requirements on any LLM serving platform.

Designing a secure, high-performance, and distributed LLM inference system requires overcoming several tightly intertwined and non-trivial challenges, each of which plays a critical role in ensuring the system's overall effectiveness. The successful execution of LLM inference in a distributed environment depends on the interplay between security, performance, and scalability, all of which need to be addressed in a seamless manner. The following challenges highlight the key areas that must be tackled to build a reliable and efficient distributed LLM inference system.

- **Inter-node Trust Establishment:** A fundamental challenges in a distributed LLM inference system is establishing mutual trust among participating nodes before exchanging sensitive information. In such setting, nodes may share user-specific data—including KV cache entries and intermediate embeddings—beyond merely model parameters. In the absence of a verifiable trust mechanism, compromised or or malicious nodes can impersonate peers, manipulate inference states, or exfiltrate private data.These risks are exacerbated in cross-domain deployments, where nodes span heterogeneous administrative boundaries and are exposed to threats such as man-in-the-middle or spoofing attacks. The lack of a centralized trust authority further complicates secure coordination. Therefore, establishing decentralized and verifiable trust relationship between nodes are essential. This trust mechanism are foundational to ensure authenticity, confidentially, and integrity across the inference pipeline.
- **Intra-node Isolation and Protection:** Even within trusted nodes, security risks persist due to untrusted co-resident workloads or adversarial software. The LLM inference process exposes privacy-sensitive artifacts—such as user prompts, attention maps, and generated tokens—that are vulnerable to memory scraping, privilege escalation, or side-channel leakage. In these context,

enforcing strong intra-node isolation is critical. Trusted Execution Environments (TEEs) offer a promising solution to these issues by providing an isolated environment for sensitive computation. However, while TEEs offer strong security guarantees, they come with certain limitations, particularly in terms of memory capacity, I/O bandwidth, and programming constraints. For resource-intensive applications like LLM inference, ensuring that the TEEs can operate efficiently while maintaining strong security guarantees is a major challenge. The balance between securing the model execution and managing the resource constraints of TEEs is essential to maintain high-performance inference while safeguarding sensitive data.

- **Secure and Low-Overhead Communication:** In a distributed LLM inference system, the communication between nodes must be secure to ensure the protection of sensitive data such as user inputs, intermediate results, and cached context embeddings. As these communications often involve highly sensitive user information, strong encryption and authentication mechanisms are necessary to prevent unauthorized access, eavesdropping, or data manipulation. However, implementing robust security protocols should not come at the expense of system performance. Low-latency communication is essential in real-time LLM inference systems to ensure fast response times and high throughput. The challenge, therefore, lies in finding a balance between securing data transmission and minimizing the performance overhead associated with encryption and authentication. This requires the design of lightweight communication protocols that can maintain the security of data exchanges without introducing significant delays. The efficiency of these protocols becomes even more critical as the scale of the distributed system grows, as each additional node introduces more potential points of communication and, consequently, greater risk of performance bottlenecks.
- **Secure and Efficient Transmission of Distributed KV Cache:** A critical component of distributed LLM inference is the efficient management and sharing of KV caches across multiple nodes. The KV cache plays a pivotal role in optimizing inference performance by storing intermediate states such as attention-key (K) and attention-value (V) embeddings. These caches are essential for maintaining context across multiple inference steps, especially in long-running or multi-turn interactions. However, since KV caches may contain sensitive user data or context from previous interactions, their secure transmission between nodes is paramount. Without proper protection, there is a risk of data leakage during the transfer of KV cache entries. Ensuring that the KV cache data is encrypted and transmitted securely across distributed nodes while maintaining performance is a delicate balance. The process of encrypting and securely transmitting KV cache entries must not introduce significant delays or computational overhead, as this would undermine the overall performance of the system. Efficient serialization, lightweight encryption, and optimized transmission protocols are necessary to ensure that KV cache data is transmitted swiftly and securely, enabling continuous context sharing across nodes while upholding strong privacy guarantees. This challenge is particularly important because the KV cache is a key performance optimization for LLM inference, and any inefficiency in its secure transmission could significantly degrade the system's overall performance.

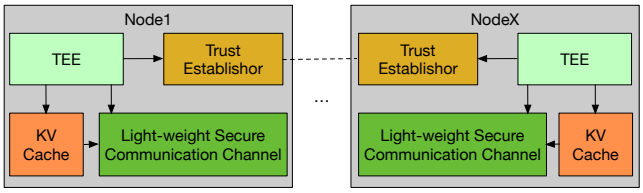

Fig. 1: Overview of the SPADA architecture

D. Threat Model

The threat model for distributed LLM inference focuses on protecting sensitive data and model states from a range of potential security risks, including unauthorized access, eavesdropping, and data tampering. Key threats include malicious or compromised nodes attempting to intercept or alter intermediate results [40], leakage of sensitive user inputs via unprotected KV caches [28], and attacks on the communication channels between nodes [25], [31], [43]. To mitigate these risks, the system employs encryption and authentication for secure data transmission, TEEs to isolate model parameters and data, and decentralized trust protocols to ensure secure, mutually authenticated node interactions. These measures aim to maintain data confidentiality, integrity, and privacy while ensuring robust performance in distributed settings.

III. System Design

To meet the stringent requirements of secure, performant, and distributed LLM inference, we propose **SPADA**—a **S**ecure, **P**erformant, and **D**istributed **A**rchitecture that tightly integrates TEEs, decentralized trust mechanisms, and optimized, enclave-aware communication protocols. SPADA is designed to mitigate the unique threats and performance constraints introduced by distributing large-scale LLM inference across potentially untrusted infrastructure. This section presents the core components of SPADA, each addressing a major challenge identified in secure and scalable LLM inference: trust establishment among nodes, intra-node protection, secure inter-node communication, and privacy-preserving KV cache transfer, shown in figure 1.

A. Decentralized Inter-node Trust Establishment

Distributed inference pipelines rely on the frequent exchange of sensitive intermediate states—such as KV cache entries, embeddings, and token predictions—across physically or administratively disparate nodes. In such environments, establishing mutual trust is a prerequisite for secure collaboration. Traditional centralized trust models based on PKI

or certificate authorities are ill-suited for federated or elastic deployments. SPADA introduces a **Decentralized Trust Establishment Protocol (DTEP)** that leverages TEE-backed remote attestation to verify node authenticity without relying on central infrastructure.

Each node in SPADA is provisioned with a hardware-backed attestation capability (e.g., Intel SGX's EPID or ECDSA attestation, AMD SEV's attestation report), which cryptographically proves the integrity of its runtime and workload to remote peers. During protocol bootstrapping, participating nodes exchange signed attestation evidence and verify each other's enclave identity, configuration, and measurement hash. This process establishes a shared root of trust across nodes operating under different domains or cloud providers. Using enclave-internal Diffie-Hellman key exchange, SPADA securely derives symmetric session keys that are never exposed outside the enclave. This secure key material underpins all subsequent data transfers and prevents unauthorized nodes from participating in the distributed inference graph.

B. Intra-node Isolation and TEE-based Protection

Even with secure inter-node communication, inference workloads remain vulnerable to a wide spectrum of intra-node threats—including privilege escalation, memory snooping, and malicious co-tenants. To mitigate these risks, SPADA leverages TEEs to enforce **strong isolation guarantees within each node**, safeguarding sensitive inference artifacts such as user inputs, attention scores, and generation history from other co-located processes.

SPADA encapsulates the entire model execution pipeline—including token embedding, Transformer blocks computations, and KV cache management—within a hardware-isolated enclave. This design ensures that adversaries with root or hypervisor-level access cannot inspect or tamper with internal inference states. However, TEEs inherently impose strict constraints on memory footprint, execution models, and I/O access due to their limited secure resources.

To address these challenges, SPADA incorporates a suite of memory- and compute-efficient techniques: quantized operator pipelines reduce model size, activation checkpointing mitigates memory pressure, and enclave-local cache eviction policies manage the constrained secure memory effectively. Additionally, SPADA employs batched enclave calls and asynchronous execution pipelines to reduce context switch overhead and maximize parallelism. Together, these optimizations enable SPADA to deliver low-latency, privacy-preserving inference while operating within the limitations of secure enclaves.

C. Secure and Low-Overhead Inter-node Communication

LLM inference is particularly sensitive to communication latency due to its sequential generation pattern and the size of intermediate state, especially in decoder-only Transformers. SPADA addresses this with a **secure and latency-optimized communication layer** that integrates cryptographic protections directly into the enclave execution flow.

SPADA terminates TLS sessions inside the enclave using enclave-compatible libraries such as WolfSSL or Rustls-TEE, ensuring that both session keys and plaintext payloads are confined within trusted boundaries. Unlike traditional models where TLS termination occurs in untrusted user space or OS layers, SPADA's approach eliminates a wide class of man-in-the-middle and key-exfiltration attacks. To reduce communication overhead, SPADA employs a binary protocol with fixed-length headers and aligned memory buffers, enabling zero-copy data transfer from enclave memory to the network stack. This design avoids unnecessary serialization or memory duplication that would otherwise incur overhead within the constrained TEE memory region. Furthermore, to defend against traffic analysis, SPADA pads messages and aggregates them into burst-mode packets, obfuscating timing and payload size correlations.

D. Secure and Efficient Distributed KV Cache Transmission

At the core of Transformer-based LLM inference lies the KV cache, which stores past attention keys and values and enables the model to process each token in constant time during decoding. In a distributed execution model, where attention heads or layers are partitioned across nodes, transmitting KV cache blocks becomes essential. However, most of these caches come from user' prompt, which contains highly sensitive informations, including user privacy information, system prompts, etc., which must be protected during transmission.

SPADA introduces a **secure and bandwidth-efficient cache propagation mechanism** that encrypts all KV cache fragments using per-session keys derived during mutual attestation. Each transmitted segment is appended with a unique nonce, sequence number, and MAC, enabling strict detection of tampering and replay attacks. To reduce communication volume, SPADA incorporates delta encoding of KV cache blocks, transmitting only the subset of keys and values that have changed between decoding steps. This is particularly effective for sparse updates common in multi-turn conversations and autoregressive decoding. Additionally, SPADA applies token-level attention predictions to preemptively prefetch relevant cache blocks, overlapping communication with computation and reducing perceived latency. Together, these optimizations allow SPADA to securely propagate cache state with minimal impact on throughput, even as context lengths scale into thousands of tokens.

IV. Conclusion

The rising computational demands of LLM inference have made distributed execution a necessity. Yet, this distribution expands the attack surface and creates an urgent need for security primitives that protect sensitive user data and model states without compromising performance. In this work, we introduced SPADA, a secure, performant, distributed inference architecture that integrates trusted execution environments, decentralized trust protocols, and low-overhead secure communication. Through mechanisms—including mutual attestation, enclave-based model execution, encrypted cache pipelines,

and delta-aware KV cache transmission—SPADA achieves strong privacy guarantees while maintaining responsiveness for interactive applications. While implementation remains ongoing, SPADA lays the groundwork for a scalable approach to secure LLM inference, paving the way for trustworthy deployment of large-scale AI models in federated, heterogeneous environments.

REFERENCES

[1] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5253–5270.

[2] S. Chen, R. Jiang, D. Yu, J. Xu, M. Chao, F. Meng, C. Jiang, W. Xu, and H. Liu, "Kvdirect: Distributed disaggregated llm inference," *arXiv preprint arXiv:2501.14743*, 2024.

[3] deepmind, "Gomini," 2025, https://deepmind.google/technologies/gemini/.

[4] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[5] L. Gao, J. Liu, H. Xu, and L. Huang, "Collaborative speculative inference for efficient llm inference serving," *arXiv preprint arXiv:2503.10325*, 2025.

[6] W. Gill, M. Elidrisi, P. Kalapatapu, A. Anwar, and M. A. Gulzar, "Privacy-aware semantic cache for large language models," *arXiv e-prints*, pp. arXiv–2403, 2024.

[7] Github, "Github copilot · your ai pair programmer," 2025, https://github.com/features/copilot/.

[8] J. Gong, V. Voskanyan, P. Brookes, F. Wu, W. Jie, J. Xu, R. Giavrimis, M. Basios, L. Kanthan, and Z. Wang, "Language models for code optimization: Survey, challenges and future directions," *arXiv preprint arXiv:2501.01277*, 2025.

[9] M. Hassid, T. Remez, J. Gehring, R. Schwartz, and Y. Adi, "The larger the better? improved llm code-generation via budget reallocation," *arXiv preprint arXiv:2404.00725*, 2024.

[10] P. He, S. Zhou, C. Li, W. Huang, W. Yu, D. Wang, C. Meng, and S. Gui, "Distributed inference performance optimization for llms on cpus," *arXiv preprint arXiv:2407.00029*, 2024.

[11] Y. He, M. Xu, J. Wu, W. Zheng, K. Ye, and C. Xu, "Uellm: A unified and efficient approach for llm inference serving," *arXiv preprint arXiv:2409.14961*, 2024.

[12] H. Heyen, A. Widdicombe, N. Y. Siegel, M. Perez-Ortiz, and P. Treleaven, "The effect of model size on llm post-hoc explainability via lime," *arXiv preprint arXiv:2405.05348*, 2024.

[13] M. Hosseinzadeh and H. Khamfroush, "Dilemma: Joint llm quantization and distributed llm inference over edge computing systems," *arXiv preprint arXiv:2503.01704*, 2025.

[14] T. Jiang, Z. Wang, J. Liang, C. Li, Y. Wang, and T. Wang, "Robustkv: Defending large language models against jailbreak attacks via kv eviction," *arXiv preprint arXiv:2410.19937*, 2024.

[15] R. Karanjai and W. Shi, "Trusted llm inference on the edge with smart contracts," in *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2024, pp. 1–7.

[16] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.

[17] B. Li, Y. Jiang, V. Gadepally, and D. Tiwari, "Llm inference serving: Survey of recent advances and opportunities," *arXiv preprint arXiv:2407.12391*, 2024.

[18] H. Li, Y. Li, A. Tian, T. Tang, Z. Xu, X. Chen, N. Hu, W. Dong, Q. Li, and L. Chen, "A survey on large language model acceleration based on kv cache management," *arXiv preprint arXiv:2412.19442*, 2024.

[19] E. Mathew, "Enhancing security in large language models: A comprehensive review of prompt injection attacks and defenses," *Authorea Preprints*, 2024.

[20] openai, "Chatgpt," 2025, https://openai.com/chatgpt/.

[21] Z. Pang, W. Wang, and Y. Liao, "Cache partitioning for mitigating timing side-channel attacks in llm serving systems," in *2024 6th International Conference on Frontier Technologies of Information and Computer (ICFTIC)*. IEEE, 2024, pp. 1238–1245.

[22] P. Patel, E. Choukse, C. Zhang, A. Shah, Í. Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient generative llm inference using phase splitting," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024, pp. 118–132.

[23] L. Qin, Q. Chen, Y. Zhou, Z. Chen, Y. Li, L. Liao, M. Li, W. Che, and P. S. Yu, "A survey of multilingual large language models," *Patterns*, vol. 6, no. 1, 2025.

[24] D. Rathee, D. Li, I. Stoica, H. Zhang, and R. Popa, "Mpc-minimized secure llm inference," *arXiv preprint arXiv:2408.03561*, 2024.

[25] L. Song, Z. Pang, W. Wang, Z. Wang, X. Wang, H. Chen, W. Song, Y. Jin, D. Meng, and R. Hou, "The early bird catches the leak: Unveiling timing side channels in llm serving systems," *arXiv preprint arXiv:2409.20002*, 2024.

[26] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, "Dynamollm: Designing llm inference clusters for performance and energy efficiency," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 1348–1362.

[27] N. Subramani, S. Luccioni, J. Dodge, and M. Mitchell, "Detecting personal information in training corpora: an analysis," in *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, 2023, pp. 208–220.

[28] S. B. Tete, "Threat modelling and risk analysis for large language model (llm)-powered applications," *arXiv preprint arXiv:2406.11007*, 2024.

[29] S. Wang, Y. Zhao, Z. Liu, Q. Zou, and H. Wang, "Sok: Understanding vulnerabilities in the large language model supply chain," *arXiv preprint arXiv:2502.12497*, 2025.

[30] B. Wu, Y. Zhong, Z. Zhang, S. Liu, F. Liu, Y. Sun, G. Huang, X. Liu, and X. Jin, "Fast distributed inference serving for large language models," *arXiv preprint arXiv:2305.05920*, 2023.

[31] G. Wu, Z. Zhang, Y. Zhang, W. Wang, J. Niu, Y. Wu, and Y. Zhang, "I know what you asked: Prompt leakage via kv-cache sharing in multi-tenant llm serving," in *Proceedings of the 2025 Network and Distributed System Security (NDSS) Symposium. San Diego, CA, USA*, 2025.

[32] Y. Xiong, J. Huang, W. Huang, X. Yu, E. Li, Z. Ning, J. Zhou, L. Zeng, and X. Chen, "High-throughput llm inference on heterogeneous clusters," *arXiv preprint arXiv:2504.15303*, 2025.

[33] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of large language models (llms): A survey," *arXiv preprint arXiv:2403.05156*, 2024.

[34] Z. Yang, Y. Yang, C. Zhao, Q. Guo, W. He, and W. Ji, "Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services," *arXiv preprint arXiv:2405.14636*, 2024.

[35] J. Zhan, W. Zhang, Z. Zhang, H. Xue, Y. Zhang, and Y. Wu, "Portcullis: A scalable and verifiable privacy gateway for third-party llm inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 1022–1030.

[36] B. Zhang, Z. Liu, C. Cherry, and O. Firat, "When scaling meets llm finetuning: The effect of data, model and finetuning method," *arXiv preprint arXiv:2402.17193*, 2024.

[37] K. Zhang, H. He, S. Song, J. Zhang, and K. B. Letaief, "Communication-efficient distributed on-device llm inference over wireless networks," *arXiv preprint arXiv:2503.14882*, 2025.

[38] K. Zhang, H. He, S. Song, J. Zhang, and K. B. Letaief, "Distributed on-device llm inference with over-the-air computation," *arXiv preprint arXiv:2502.12559*, 2025.

[39] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett *et al.*, "H2o: Heavy-hitter oracle for efficient generative inference of large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 34 661–34 710, 2023.

[40] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[41] J. Zhao, Y. Song, S. Liu, I. G. Harris, and S. A. Jyothi, "Lingualinked: A distributed large language model inference system for mobile devices," *arXiv preprint arXiv:2312.00388*, 2023.

[42] J. Zhao, Y. Song, S. Liu, I. G. Harris, and S. A. Jyothi, "Lingualinked: Distributed large language model inference on mobile devices," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 2024, pp. 160–171.

[43] X. Zheng, H. Han, S. Shi, Q. Fang, Z. Du, X. Hu, and Q. Guo, "Inputsnatch: Stealing input in llm services via timing side-channel attacks," *arXiv preprint arXiv:2411.18191*, 2024.
[44] Z. Zheng, X. Ji, T. Fang, F. Zhou, C. Liu, and G. Peng, "Batchllm: Optimizing large batched llm inference with global prefix sharing and throughput-oriented token batching," *arXiv preprint arXiv:2412.03594*, 2024.
[45] I. Zimerman, A. Adir, E. Aharoni, M. Avitan, M. Baruch, N. Drucker, J. Lerner, R. Masalha, R. Meiri, and O. Soceanu, "Power-softmax: Towards secure llm inference over encrypted data," *arXiv preprint arXiv:2410.09457*, 2024.