# Hogwild! Inference: Parallel LLM Generation via Concurrent Attention

Gleb Rodionov* [2] Roman Garipov* [1,2] Alina Shutova* [1,2]
George Yakushev* [1,2] Erik Schultheis* [3] Vage Egiazarian [3]
Anton Sinitsin [2] Denis Kuznedelev [2] Dan Alistarh [3]

[1]HSE University [2]Yandex Research [3]IST Austria

## The Sequential Inference Problem

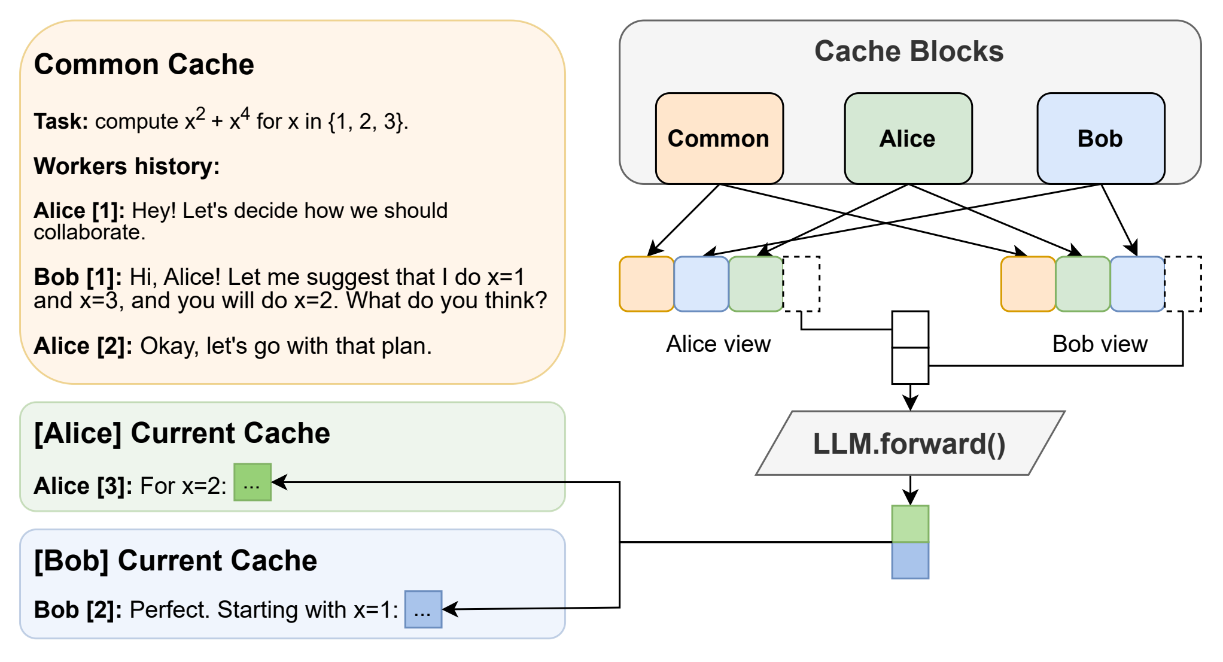
Large Language Models face a fundamental bottleneck: they generate tokens sequentially despite having access to powerful parallel hardware. This creates significant latency for complex reasoning tasks that require long chains of thought. Existing parallel approaches like Self-Consistency and Skeleton-of-Thought rely on rigid, pre-defined collaboration patterns that cannot adapt dynamically during problem-solving.

### Our Solution: Hogwild! Inference

We introduce a novel parallel inference engine where multiple LLM workers generate tokens concurrently through a shared attention cache. This enables real-time collaboration without any fine-tuning, working with modern reasoning-capable LLMs out-of-the-box. Our approach achieves up to 2.8x speedup with 4 workers while maintaining or even improving accuracy across diverse benchmarks.

## Core Technical Innovation

Hogwild! Inference enables concurrent attention through shared Key-Value memory, allowing multiple LLM instances to run in parallel while seeing each other's tokens immediately. Instead of re-computing KV representations, we efficiently stitch them together by adjusting positional embeddings via RoPE rotation.



**Common Cache**

**Task:** compute $x^2 + x^4$ for x in {1, 2, 3}.

**Workers history:**

**Alice [1]:** Hey! Let's decide how we should collaborate.

**Bob [1]:** Hi, Alice! Let me suggest that I do x=1 and x=3, and you will do x=2. What do you think?

**Alice [2]:** Okay, let's go with that plan.

**[Alice] Current Cache**

**Alice [3]:** For x=2: ...

**[Bob] Current Cache**

**Bob [2]:** Perfect. Starting with x=1: ...

Figure 1. Hogwild! Inference with 2 workers and 3 shared cache blocks. Workers see common prompts and each other's tokens through rotated positional embeddings.

### Emergent Collaborative Behaviors

Modern reasoning-capable LLMs naturally coordinate without pre-defined frameworks. Workers organically develop divide-and-conquer strategies, perform real-time error correction, and dynamically re-plan when initial approaches fail. GPT-4o analysis confirms that token-wise synchronization enables significantly better collaboration.

## Efficient Implementation

Our key technical innovation is query rotation instead of KV cache rotation. Instead of rotating all cached keys (inefficient for long contexts), we rotate only the current token queries. This leverages RoPE properties to maintain attention consistency with minimal overhead, avoiding $O(n^3)$ recomputation complexity.

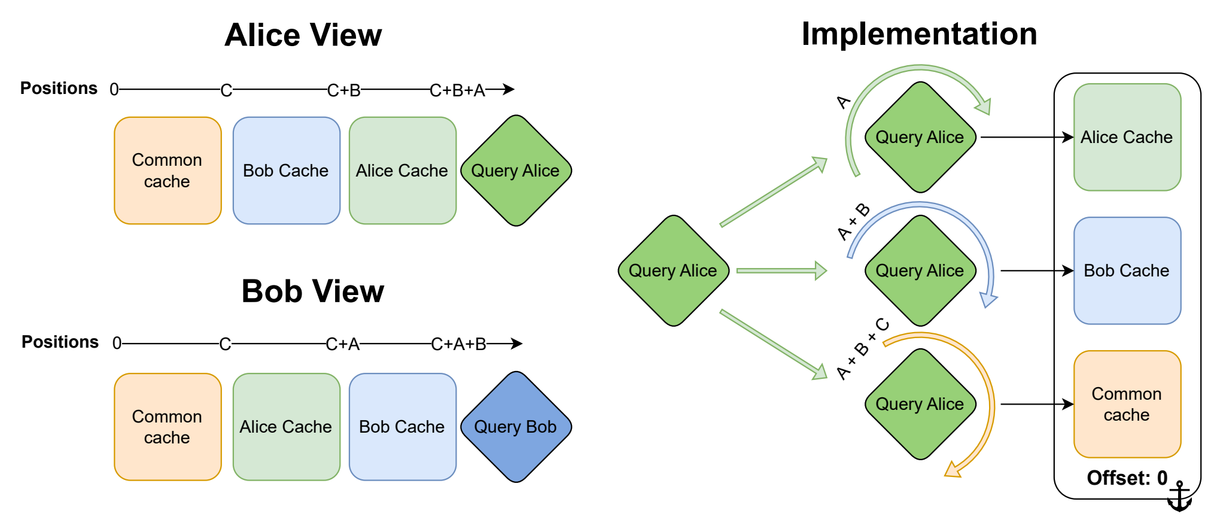


Figure 2. Inference scheme with query rotation. Instead of rotating cache blocks, we rotate current token queries to equivalent angles, enabling efficient real-time synchronization.

The implementation uses custom GPU kernels based on Flash-Decoding principles, gathering each KV cache block in contiguous memory buffers. This approach provides near-linear scaling with worker count while maintaining minimal constant latency overhead compared to baseline sequential inference.

## LIMO Mathematical Reasoning Results

On the challenging LIMO mathematical reasoning benchmark, Hogwild! Inference maintains reasoning quality while providing substantial speedup. With 2 workers, it achieves better accuracy with the same token budget and consistently outperforms Self-Consistency and Skeleton-of-Thought baselines.

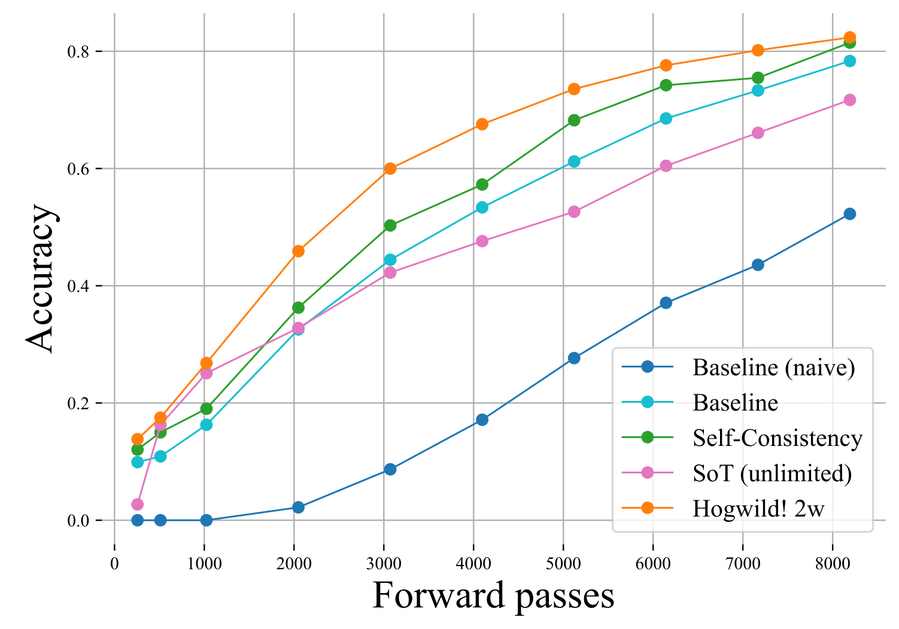


Figure 3. Hogwild! Inference converges faster to correct solutions on LIMO dataset with QwQ-32B, demonstrating effective collaboration on complex mathematical problems.

Evaluation across multiple model families including QwQ-32B, Qwen3, and Phi-4 shows consistent improvements. Workers naturally identify opportunities for parallel sub-task execution and error correction, often achieving higher final accuracy than single-worker baselines.

## LiveCodeBench Programming Evaluation

Hogwild! Inference proves effective beyond mathematical reasoning, demonstrating strong performance on LiveCodeBench programming tasks. The method improves Pass@1 across multiple model families while maintaining code quality.

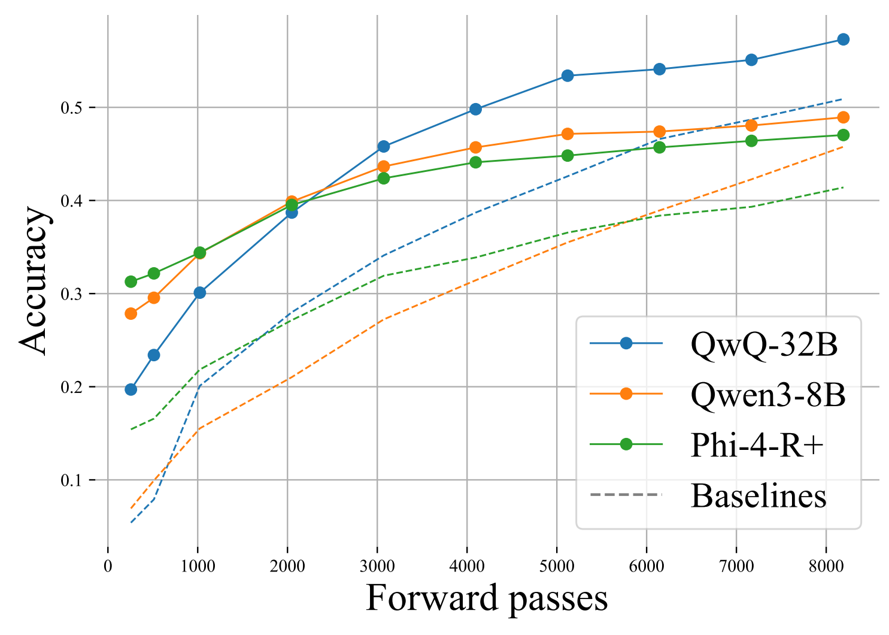


Figure 4. LiveCodeBench results showing Hogwild! improves Pass@1 across model families with 2 workers, demonstrating versatility for code generation tasks.

Workers naturally divide coding responsibilities, cross-verify implementations, and collaboratively debug solutions. This emergent collaboration results in more robust and correct code generation compared to sequential approaches.

## Performance & Scalability

Hogwild! Inference demonstrates near-linear scaling with worker count while maintaining minimal overhead. Performance benchmarks show consistent improvements across different context lengths.

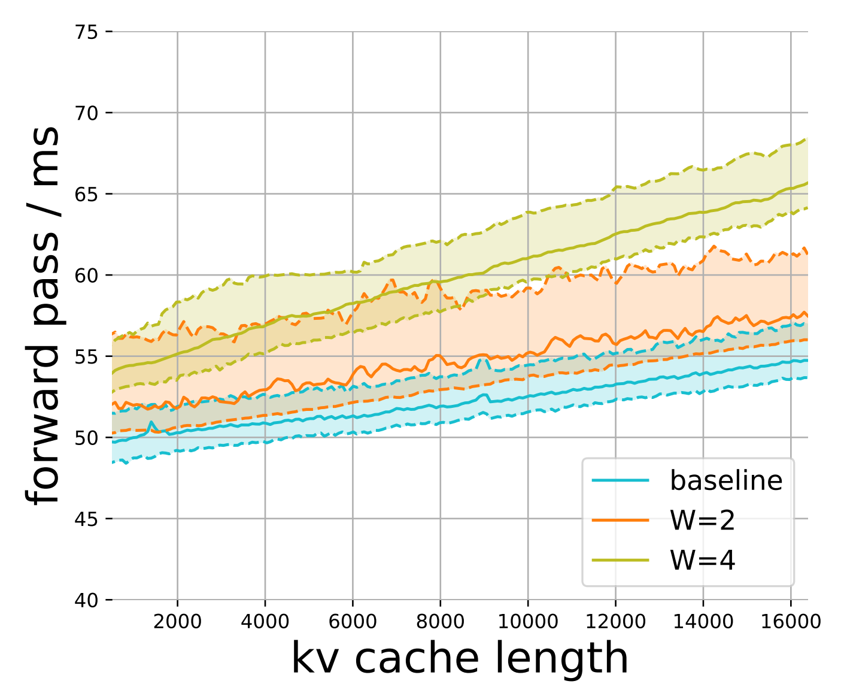


Figure 5. Single forward pass duration for QwQ-32B-AWQ shows near-linear scaling with worker count, with 2 workers providing 1.8-2.0x improvement and 4 workers achieving 3.2-3.6x.

### Key Performance Metrics:

- 2 workers: 1.8-2.0x tokens/second improvement
- 4 workers: 3.2-3.6x tokens/second improvement
- GPU utilization increases from 25% to 72%
- Works with QwQ, DeepSeek-R1, Qwen3, Phi-4 out-of-the-box
- No fine-tuning required

Code:
`https://github.com/eqimp/hogwild_llm`