# 10-718 Assignment 1: Problem Proposal

Yixin Dong and Akaash R. Parthasarathy

# Problem Description

LLMs are increasingly capable of writing complete system code

- LLMs run on serving engines   etc.
- Serving engines call GPU kernels
- GPU kernels are highly specialized
  - For both hardware and the actual workload
  - Require significant effort and expertise to optimize

**Questions:**

- Can we generate kernels that are correct and achieve strong performance on different workloads from the trace distribution?
- What kind of LLM/agent design is suitable for this task?
- Model efficiency (reducing benchmark calls and kernel generation latency)

# FlashInfer Trace Schema

- Use FlashInfer Trace Schema to study data-driven GPU kernel generation
  - Kernel **Definition**
  - **Workload** specification
  - Generate candidate **Solutions**
  - Use FlashInfer-Bench to run correctness and performance benchmarking and record results as new **Traces**

# Target Kernels

### Fused MoE

Fused Mixture-of-Experts kernels with FP8 support.

FP8 MoE

### Sparse Attention

Deepseek Sparse Attention from Deepseek V3.2

Indexer

Attention

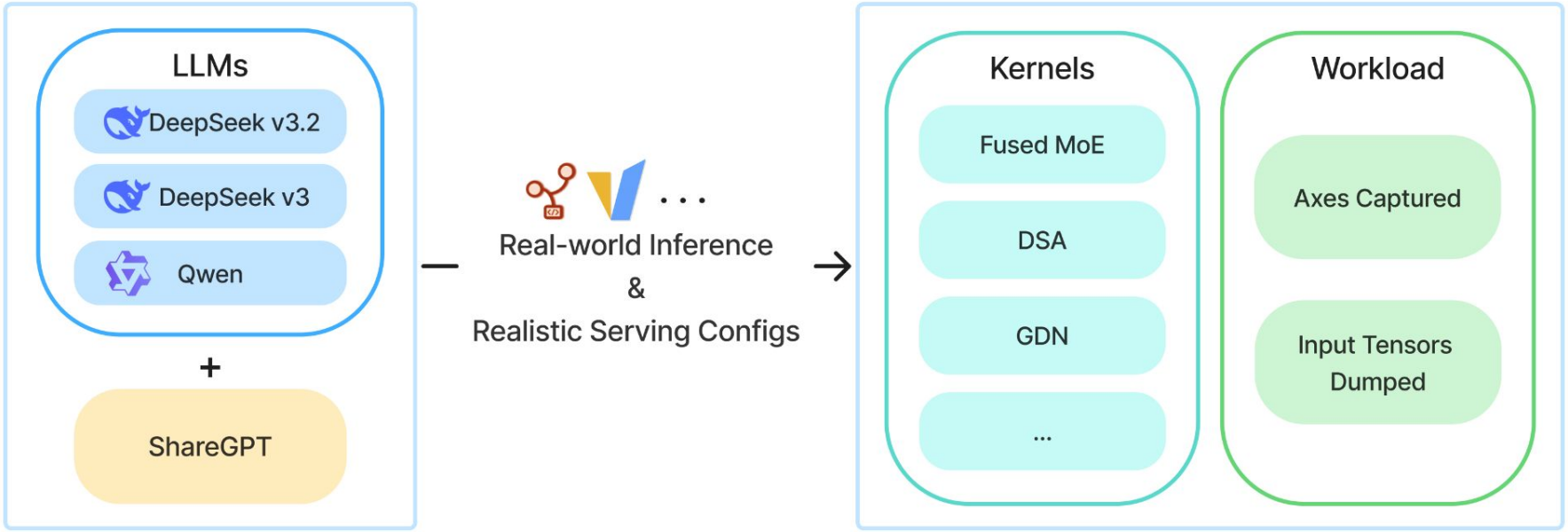
### Gated Delta Net

Gated Delta Net used in Qwen3-Next

GDN Decode

GDN Prefill

# Collection Process



# Collection Methods

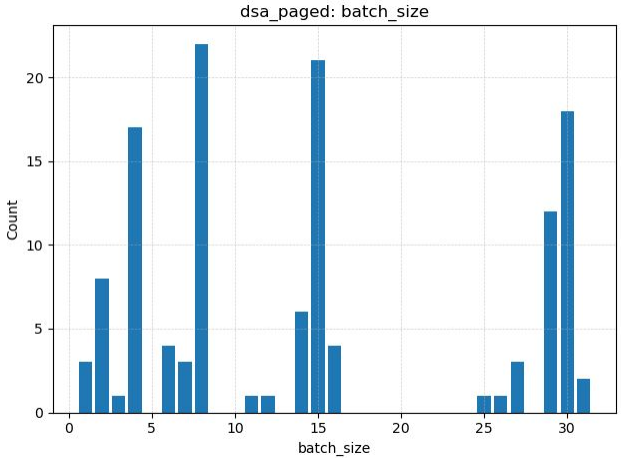
We have three curation paths (increasing levels of automation):

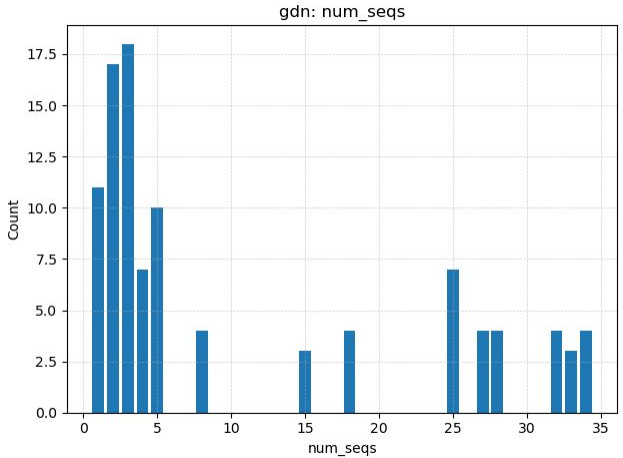
- Manually inject dump statements
  - in the execution path
- Auto-tracing via flashinfer_bench.tracing
  - needs adapter → schema
- LLM-assisted generation
  - Claude skills to write dumps/adapters

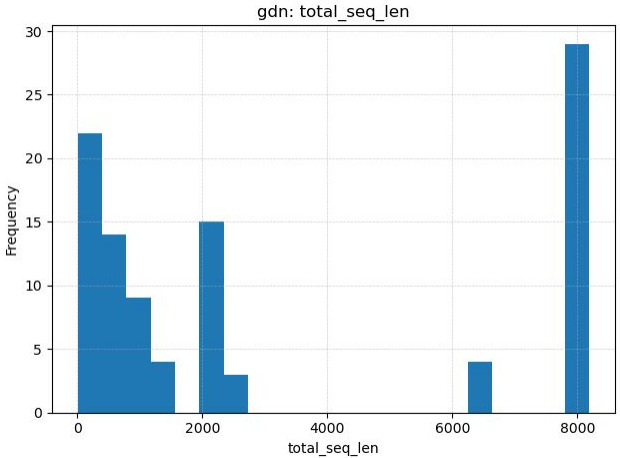
# Overview of the Collected Workloads

Workloads:

- dsa_paged: 151
- gdn: 120
- moe: 19



# Proposed Agent Design

Prompt: describe mathematical definition, basic guidelines

N Iterative Feedback Rounds -> best performing kernel

Generate Triton and CUDA kernels for B200

**Baseline:** Let LLMs directly generate a CUDA kernel with minimal prompt design

**Input** : Definition, Language, Hardware
**Output** : $\text{Sol}^*$ (best solution)
$\mathcal{S} \leftarrow \emptyset$
Agent $\leftarrow$ CodeAgent.Initialize(Definition, Language, Hardware)
$\text{Sol}_0 \leftarrow$ Agent.Generate()
**for** $i \leftarrow 0$ **to** $N-1$ **do**
  $\text{Trace}_i \leftarrow$ FlashInfer-Bench.Benchmark(Definition, $\text{Sol}_i$)
  **if** $\textit{Trace}_i.\textit{Status} = \textsc{Passed}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\text{Sol}_i, \text{Trace}_i)\}$
  $\text{Sol}_{i+1} \leftarrow$ Agent.Optimize($\text{Trace}_i$)
$\text{Sol}^* \leftarrow \arg\max_{(\text{Sol}_i, \text{Trace}_i) \in \mathcal{S}} \text{Trace}_i.\text{Speedup}$
**return** $\textit{Sol}^*$

*Thanks!*