

Oct 10, 2026

Your model deserves a better API than a prompt.

Pie: programmable serving



Felix Zou and Lin Zhong, Yale University

In training, you can write the loop.

someone else's loop

```
model.fit(x, y, epochs=10)
```

Keras

your own loop

```
for x, y in loader:  
    loss = loss_fn(model(x), y)  
    loss.backward()  
    opt.step()  
    opt.zero_grad()
```

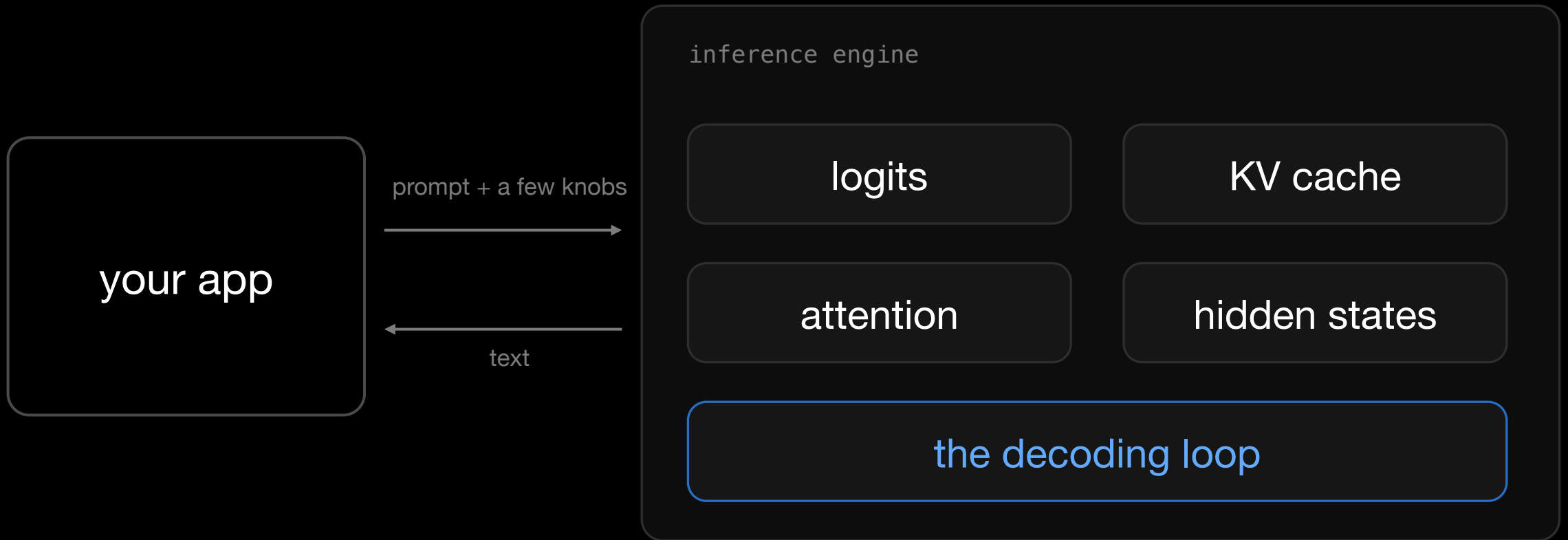
PyTorch

INFERENCE

```
client.chat.completions.create(  
    model="qwen3.6-35b-a3b",  
    messages=[{"role": "user", "content": prompt}],  
    temperature=0.7,  
)
```

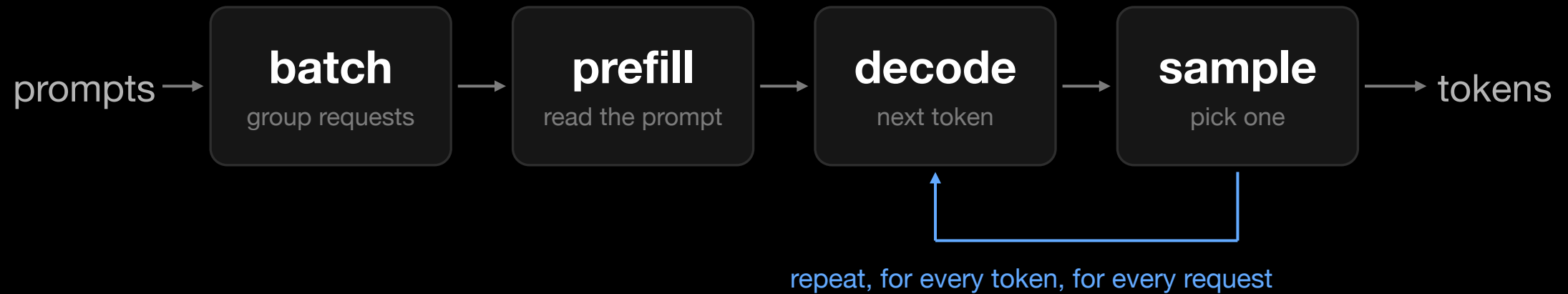
In inference, this is all you get.

Everything interesting is on the other side.



THE PROBLEM

Every engine runs one loop, for everyone.



vLLM · SGLang · llama.cpp · Ollama · LM Studio

63

global knobs in vllm serve

Set once by whoever runs the server.
Same for every request.

```
usage: vllm serve [-h] [--model MODEL] [--task {auto,classify,draft,embed,embedding,generate,reward,score,transcription}] [--tokenizer TOKENIZER] [--tokenizer-mode {auto,custom,mistral,slow}] [--trust-remote-code | --no-trust-remote-code] [--dtype {auto,bfloat16,float,float16,float32,half}] [--seed SEED] [--hf-config-path HF_CONFIG_PATH] [--allowed-local-media-path ALLOWED_LOCAL_MEDIA_PATH] [--revision REVISION] [--code-revision CODE_REVISION] [--rope-scaling ROPE_SCALING] [--rope-theta ROPE_THETA] [--tokenizer-revision TOKENIZER_REVISION] [--max-model-len MAX_MODEL_LEN] [--quantization {aqlm,awq,awq_marlin,bitblas,bitsandbytes,compressed-tensors,deepspeedfp,experts_int8,fbgemm_fp8,fp8,gguf,gptq,gptq_bitblas,gptq_marlin,gptq_marlin_24,hqq,ipex,marlin,modelopt,moe_wna16,neuron_quant,nvfp4,ptpc_fp8,qqq,quark,torchao,tpu_int8,None}] [--enforce-eager | --no-enforce-eager] [--max-seq-len-to-capture MAX_SEQ_LEN_TO_CAPTURE] [--max-logprobs MAX_LOGPROBS] [--disable-sliding-window | --no-disable-sliding-window] [--disable-cascade-attn | --no-disable-cascade-attn] [--skip-tokenizer-init | --no-skip-tokenizer-init] [--enable-prompt-embeds | --no-enable-prompt-embeds] [--served-model-name SERVED_MODEL_NAME [SERVED_MODEL_NAME ...]] [--disable-async-output-proc] [--config-format {auto,hf,mistral}] [--hf-token [HF_TOKEN]] [--hf-overrides HF_OVERRIDES] [--override-neuron-config OVERRIDE_NEURON_CONFIG] [--override-pooling OVERRIDE_POOLING] [--override-processor-pattern PROCESSOR_PATTERN] [--generation-config GENERATION_CONFIG] [--override-generation-config OVERRIDE_GENERATION_CONFIG] [--load-format {auto,vllm,transformers}] [--load-format-extra-kwargs LOAD_FORMAT_EXTRA_KWARGS] [--download-dir DOWNLOAD_DIR] [--model-load-tqdm-on-load | --no-use-tqdm-on-load] [--guided-decoding-backend {auto,guidance,openai} | --guided-decoding-backend-fallback] [--guided-decoding-disable-async | --no-guided-decoding-disable-async] [--distributed-executor-backend {external,ray} | --distributed-executor-backend-tensor-parallel-size TENSOR_PARALLEL_SIZE] [--data-parallel-size DATA_PARALLEL_SIZE] [--data-parallel-address DATA_PARALLEL_ADDRESS] [--max-parallel-loading-workers MAX_PARALLEL_LOADING_WORKERS] [--custom-all-reduce | --no-custom-all-reduce] [--worker-cls WORKER_CLASS] [--gpu-memory-utilization GPU_MEMORY_UTILIZATION] [--num-gpu-blocks-override NUM_GPU_BLOCKS_OVERRIDE] [--enable-prefill CPU_OFFLOAD_GB] [--calculate-kv-scales] [--tokenizer-pool-type TOKENIZER_POOL_TYPE] [--tokenizer-pool-size TOKENIZER_POOL_SIZE] [--mm-processor-kwarg MM_PROCESSOR_KWARG] [--disable-mm-prefill] [--no-enable-lora-bias] [--max-loras MAX_LORAS] [--max-lora-rank MAX_LORA_RANK] [--lora-extra-vocab-size LORA_EXTRA_VOCAB_SIZE] [--lora-dtype {auto,bfloat16,float16}] [--long-lora-scaling-factors LONG_LORA_SCALING_FACTORS [LONG_LORA_SCALING_FACTORS ...]] [--max-cpu-loras MAX_CPU_LORAS] [--fully-sharded-loras | --no-fully-sharded-loras] [--enable-prompt-adapter | --no-enable-prompt-adapter] [--max-prompt-adapters MAX_PROMPT_ADAPTERS] [--max-prompt-adapter-token MAX_PROMPT_ADAPTER_TOKEN] [--device {auto,cpu,cuda,hpu,neuron,tpu,xpu}] [--speculative-config SPECULATIVE_CONFIG] [--show-hidden-metrics-for-version SHOW_HIDDEN_METRICS_FOR_VERSION] [--otlp-traces-endpoint OTLP_TRACES_ENDPOINT] [--collect-detailed-traces {all,model,worker,None} [{all,model,worker,None} ...]] [--max-num-batched-tokens MAX_NUM_BATCHED_TOKENS] [--max-num-seqs MAX_NUM_SEQS] [--max-num-partial-prefills MAX_NUM_PARTIAL_PREFILLS] [--max-long-partial-prefills MAX_LONG_PARTIAL_PREFILLS] [--cuda-graph-sizes CUDA_GRAPH_SIZES [CUDA_GRAPH_SIZES ...]] [--long-prefill-token-threshold LONG_PREFILL_TOKEN_THRESHOLD] [--num-lookahead-slots NUM_LOOKAHEAD_SLOTS] [--scheduler-delay-factor SCHEDULER_DELAY_FACTOR] [--preemption-mode {recompute,swap,None}] [--num-scheduler-steps NUM_SCHEDULER_STEPS] [--multi-step-stream-outputs | --no-multi-step-stream-outputs] [--scheduling-policy {fcfs,priority}] [--enable-chunked-prefill | --no-enable-chunked-prefill] [--disable-chunked-mm-input | --no-disable-chunked-mm-input] [--scheduler-cls SCHEDULER_CLS] [--kv-transfer-config KV_TRANSFER_CONFIG] [--kv-events-config KV_EVENTS_CONFIG] [--compilation-config COMPILE_CONFIG] [--additional-config ADDITIONAL_CONFIG] [--use-v2-block-manager] [--disable-log-stats]
```

THE PROBLEM

New idea? Wait for the maintainers.

vllm-project/vllm #6226

[RFC] Drop beam search support

July 2024

sgl-project/sglang #3032

[Feature] Support Beam Search

January 2025 · closed for inactivity

Beam search is a textbook algorithm. Now imagine yours.

What a fixed loop costs you

01

Agents ping-pong

Every tool call leaves the engine. The history gets re-read when the result comes back.

02

One policy for all

Your app knows what to cache and when to speculate. The engine can't ask.

03

Have to fork the engine

Plugins only reach the scores. Changing the loop itself means maintaining a fork.

HF transformers

the loop in your research notebook

slow · one user

↑ hackable: write your own loop



fast · many users →

vLLM · SGLang
llama.cpp · Ollama

production serving

fixed loop

HF transformers

the loop in your research notebook

slow · one user

↑ hackable: write your own loop



your loop, at full speed

fast · many users →

vLLM · SGLang
llama.cpp · Ollama

production serving

fixed loop

THE IDEA

**Serve programs,
not prompts.**

Inferlets

Small programs, in Python, JavaScript or Rust, that run inside the server, right next to the model.

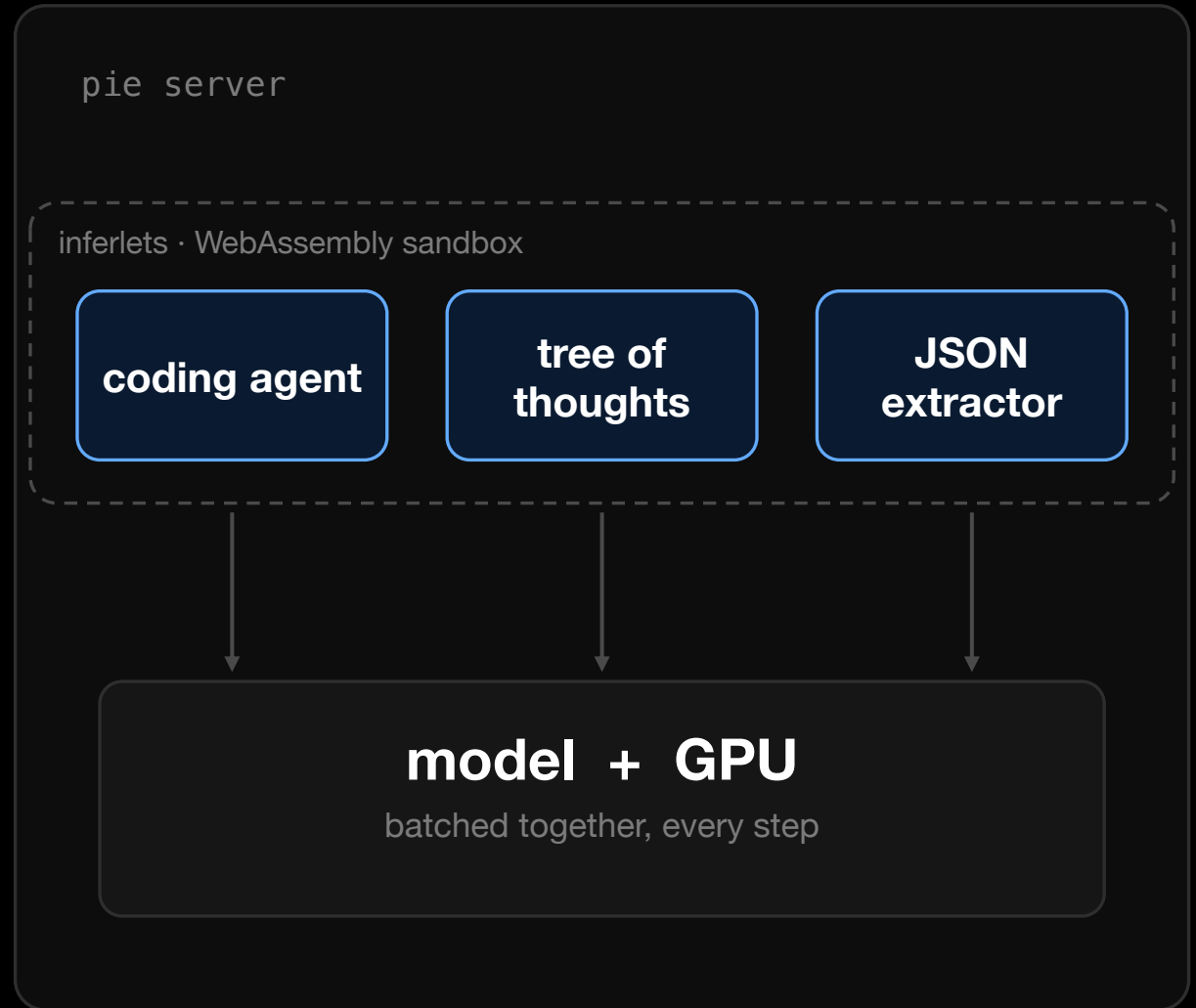
Each one owns

the loop forward passes & sampling

the memory its own KV cache

the I/O tools, HTTP, other agents

Pie still does the hard part: batching everyone, memory, GPU kernels.



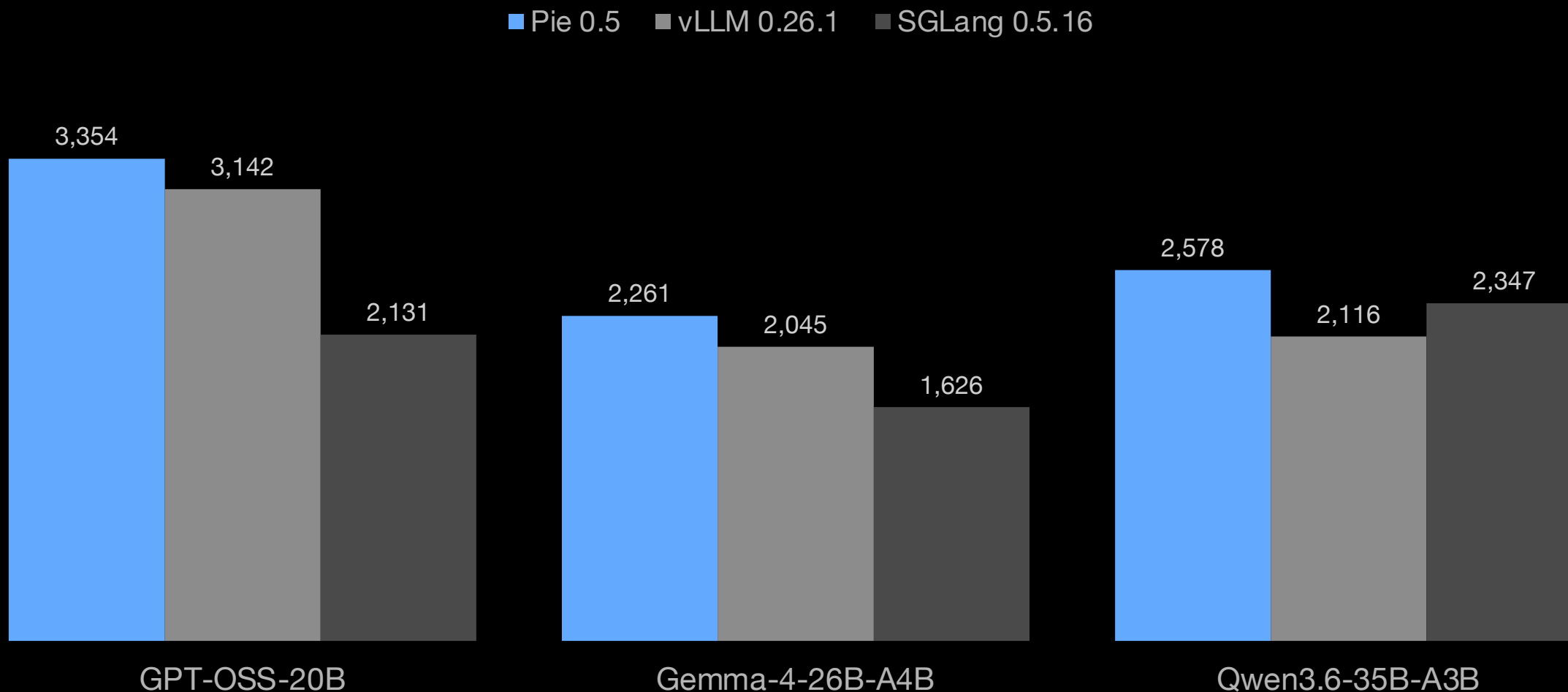
**To Pie, vLLM is just
one hard-coded inferlet.**

Pie's own OpenAI, Anthropic and Gemini APIs are inferlets, too.

QUESTION

But surely programmability costs performance?

As fast as the engines it replaces. Often faster.



Output tokens/s · 256 concurrent requests · one NVIDIA A100 · Pie 0.5 vs vLLM 0.26.1 and SGLang 0.5.16

HOW

Your loop, compiled to the GPU.

in training

```
model = torch.compile(model)
```

Your PyTorch, fused into GPU kernels.

in pie

```
fwd.epilogue(lambda: masked_argmax(logits(), mask))
```

Your sampler, traced once and fused into every GPU step.

Python plans the work. The GPU runs the loop.

in Pie: TokenFlow

32 AGENTS · ONE MACBOOK

4.7×

Ollama's throughput, same laptop, same model

355 vs 75

tokens/s across all agents

0.6 s vs 54 s

median wait for a first token

WHAT YOU CAN BUILD

So what will you build?

agents · inference-time scaling · research that ships

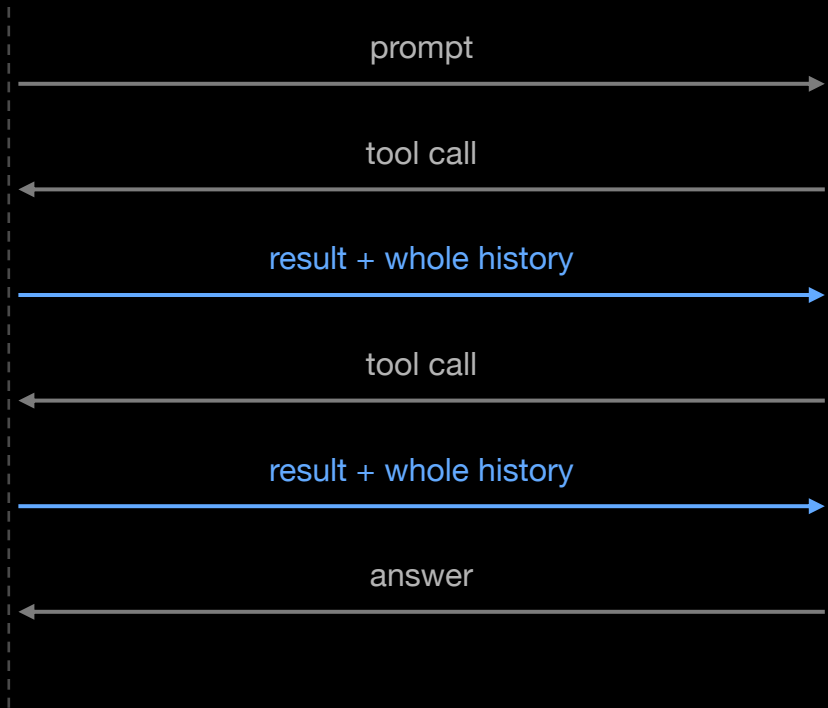
AGENTS

The tool loop runs inside the engine.

today

your app

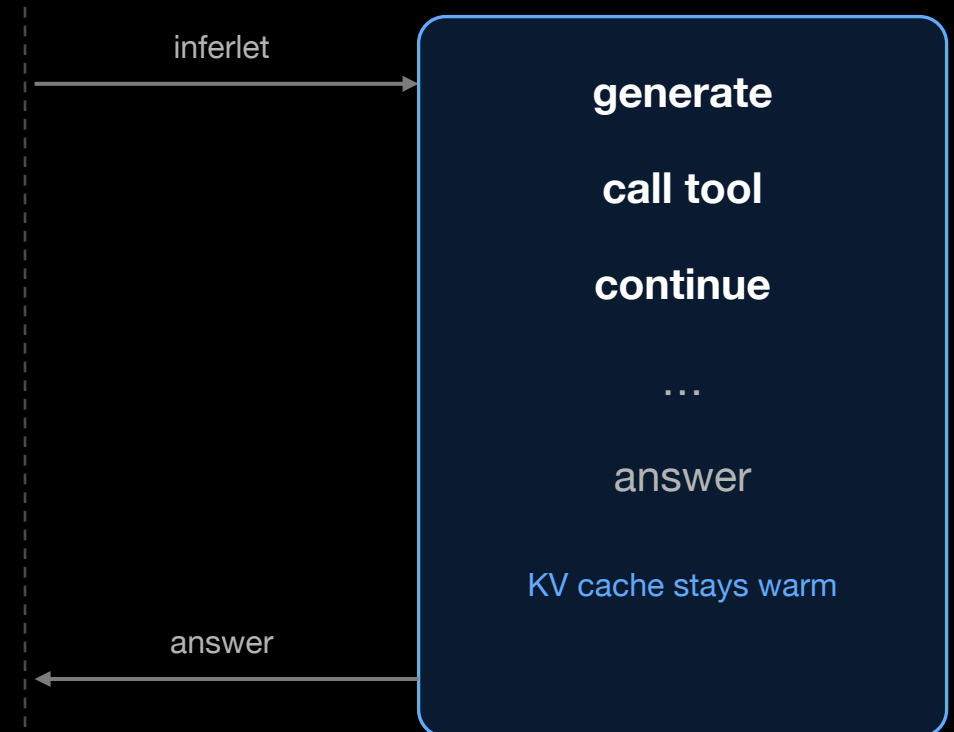
engine



with pie

your app

pie

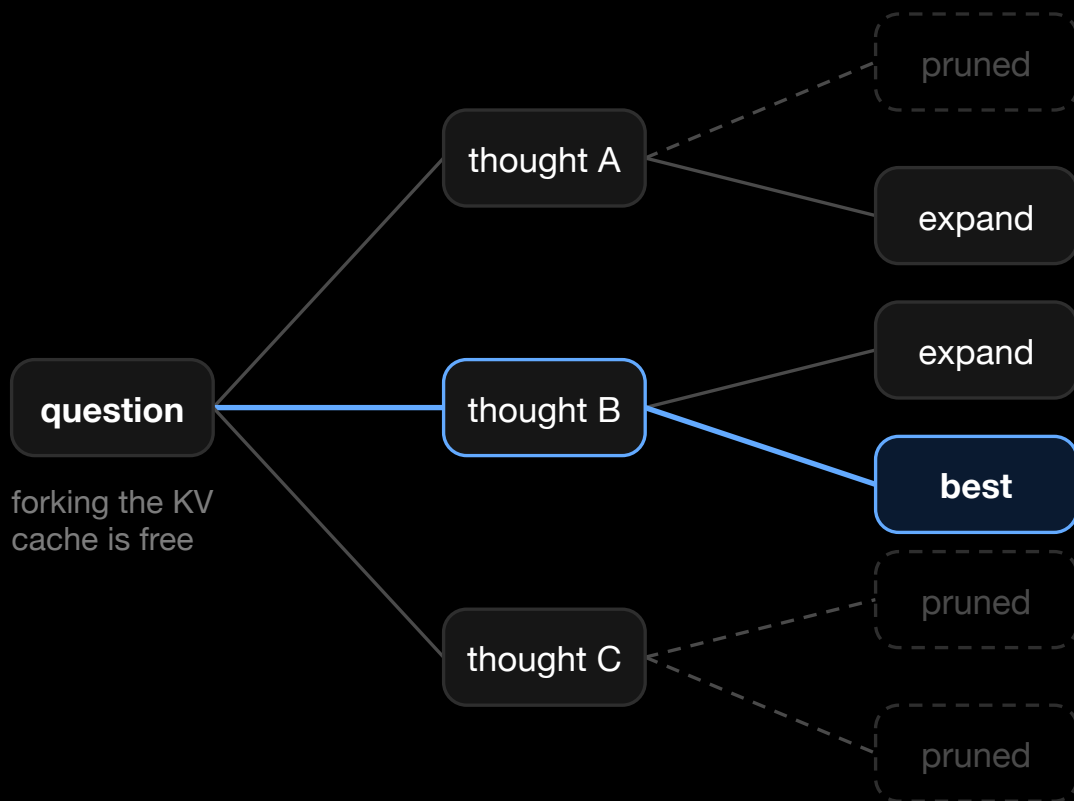


3.5x

throughput vs. vLLM on an agent workload, from three app-specific tricks:

- 1 Keep the docs of tools you call often in cache
- 2 Drop specs you'll never need again
- 3 Fire each tool the moment its call is parsed

Tree of thoughts, inside the server.



Every branch shares one KV cache.

No round trip per thought.

Your decoding paper is 10–50 lines.

SAMPLING

Mirostat
Locally typical
 η / ϵ -sampling
Tail-free
Top-a · XTC · DRY
Entropy-adaptive temp.

STEERING

Classifier-free guidance
Contrastive decoding
Context-aware decoding

SEARCH

Beam search
Self-consistency
ASAp grammar sampling

SPECULATIVE

MTP heads
DFlash block drafts
Cacheback (n-gram)
Grammar + speculation

KV CACHE

SnapKV · H2O · TOVA
Attention sinks
Sliding windows
Prefix trees

STRUCTURED

JSON schema
Token healing

WATERMARKS

SynthID
Gumbel
Green-list (KGW)

DIFFUSION

Diffusion-LLM samplers
Text → image
Text → video

USE IT TODAY

Drop-in today.

```
$ curl -fsSL pie-project.org/install.sh | bash  
$ pie model import Qwen/Qwen3.5-0.8B  
$ pie serve
```

```
OpenAI(base_url="http://localhost:8080/v1")
```

**Speaks the OpenAI,
Anthropic and
Gemini APIs.**

Your agent framework
already works. Drop down to
inferlets only where you need
control.

Runs on what you have.

HARDWARE

NVIDIA GPUs

Apple Silicon

AMD & Intel (Vulkan)

Browser (WebGPU)

iPhone

+ more coming

MODELS

Qwen 3.5 – 3.8

Gemma 4

gpt-oss

DeepSeek-V4

GLM-5

FLUX.2 · image

Wan 2.2 · video

+ more added regularly

Apache 2.0 · inferlets in Python, JavaScript or Rust · pre-release, moving fast

Go write your own loop.

github.com/pie-project/pie

Next: a local app built on Pie →

