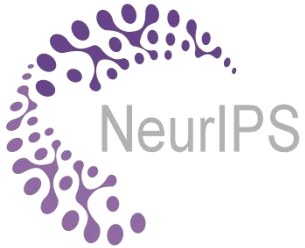


KVsail: Cross-request KV cache with Session Management and Dynamic Offloading for Large Language Model Serving

Tianyu Guo†, Hande Dong*, Yichong Leng★, Feng Liu*, Qiang Lin*, Xianwei Zhang†

†Sun Yat-Sen University

*Tencent Inc

★University of Science and Technology of China

用户和大语言模型多轮交互

- 用户会与大语言模型发送多轮提示词
- 每一轮的交互会附带上下文信息
- 每一轮的上下文信息存在重复的前缀

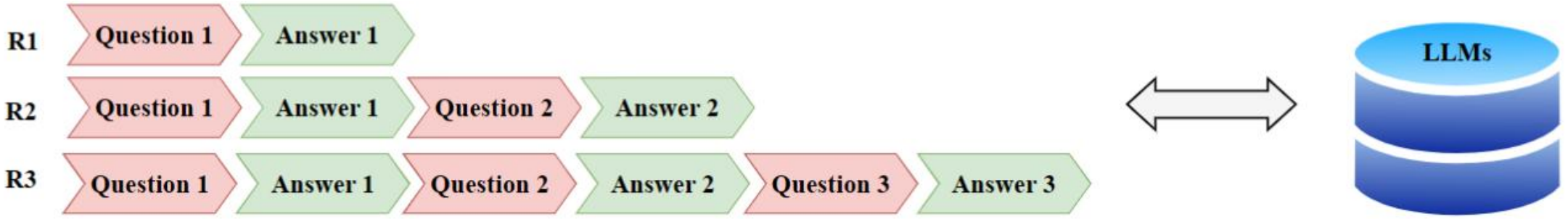
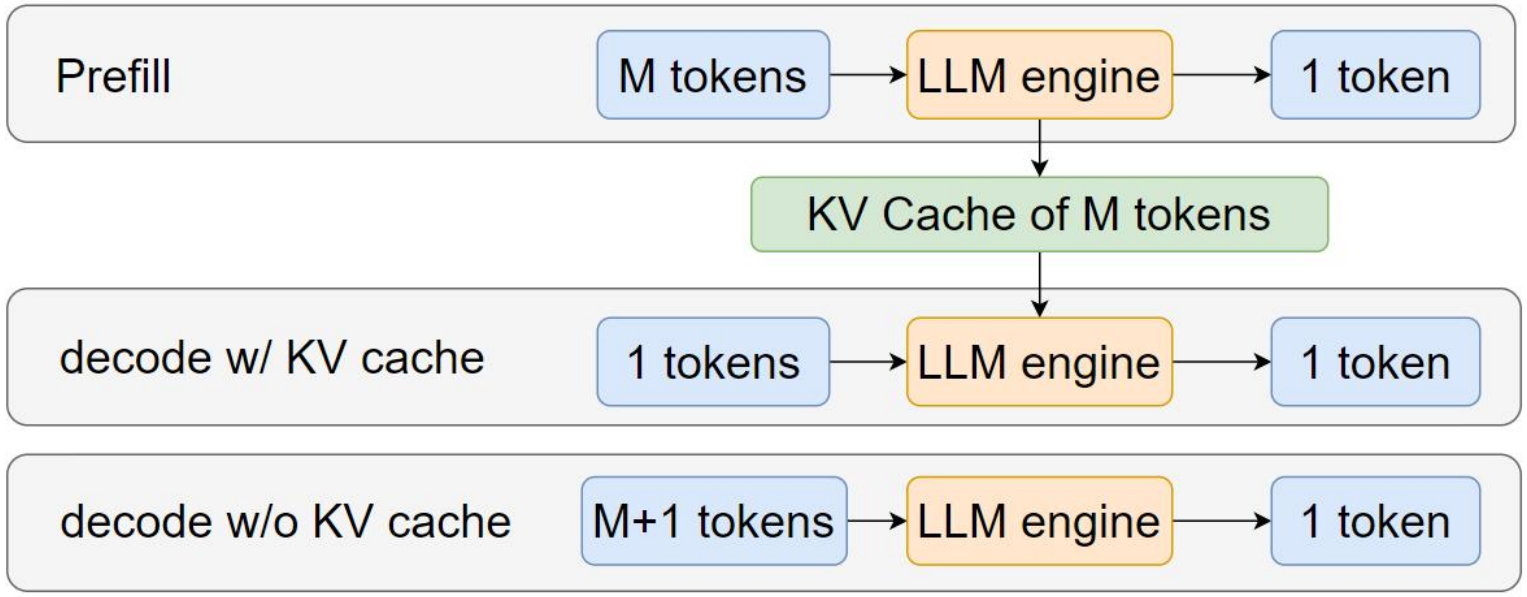


Figure 1: Illustrative example of multi-turn dialogue (where *R* stands for *Round*).

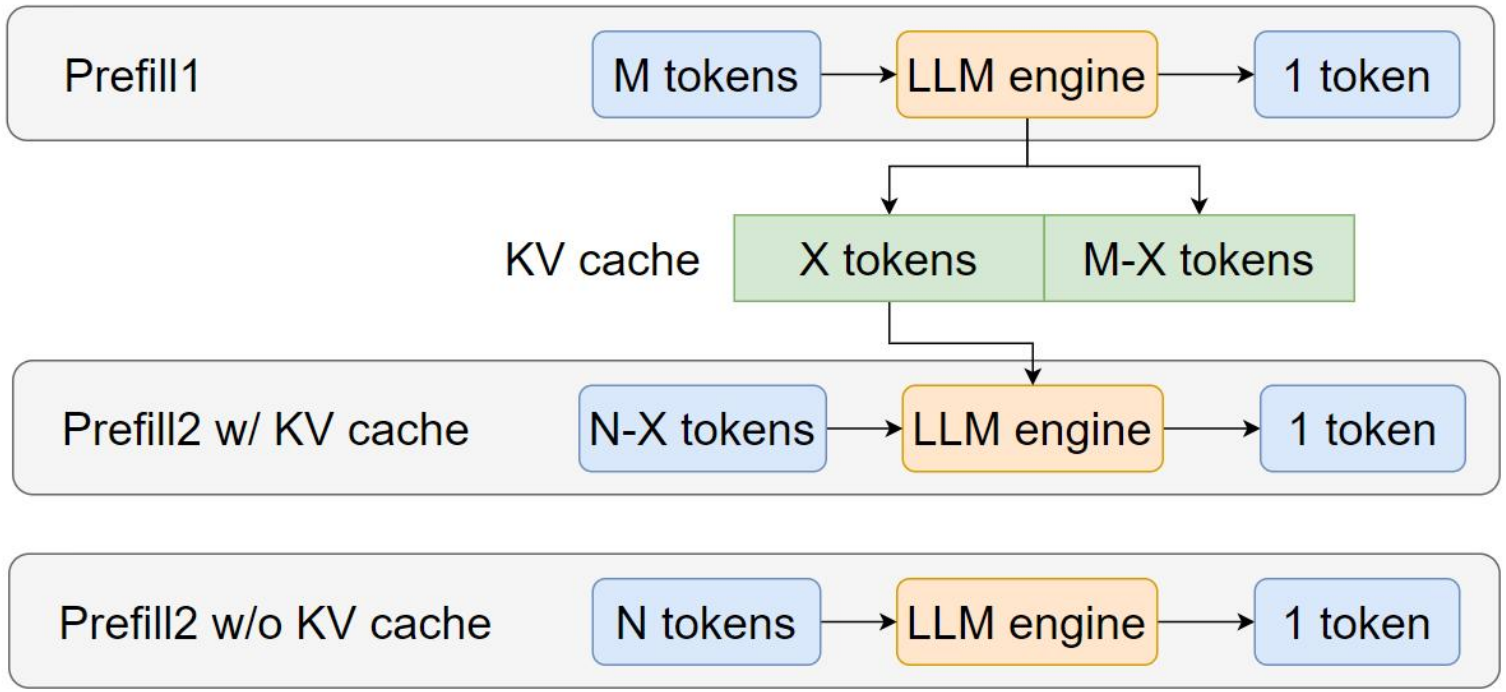
大语言模型推理的中间状态

- KV缓存保存了大语言模型推理的中间状态（上下文信息）
- KV缓存可以加速大语言模型的推理过程



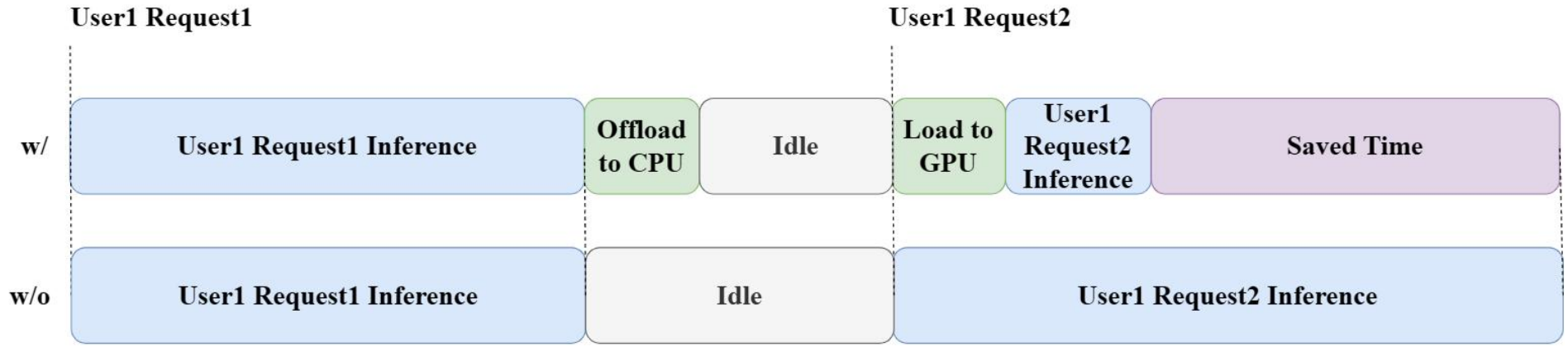
跨请求KV缓存复用

- 前缀相同的序列KV缓存是相同的
- context prefill：有KV缓存输入的Prefill阶段



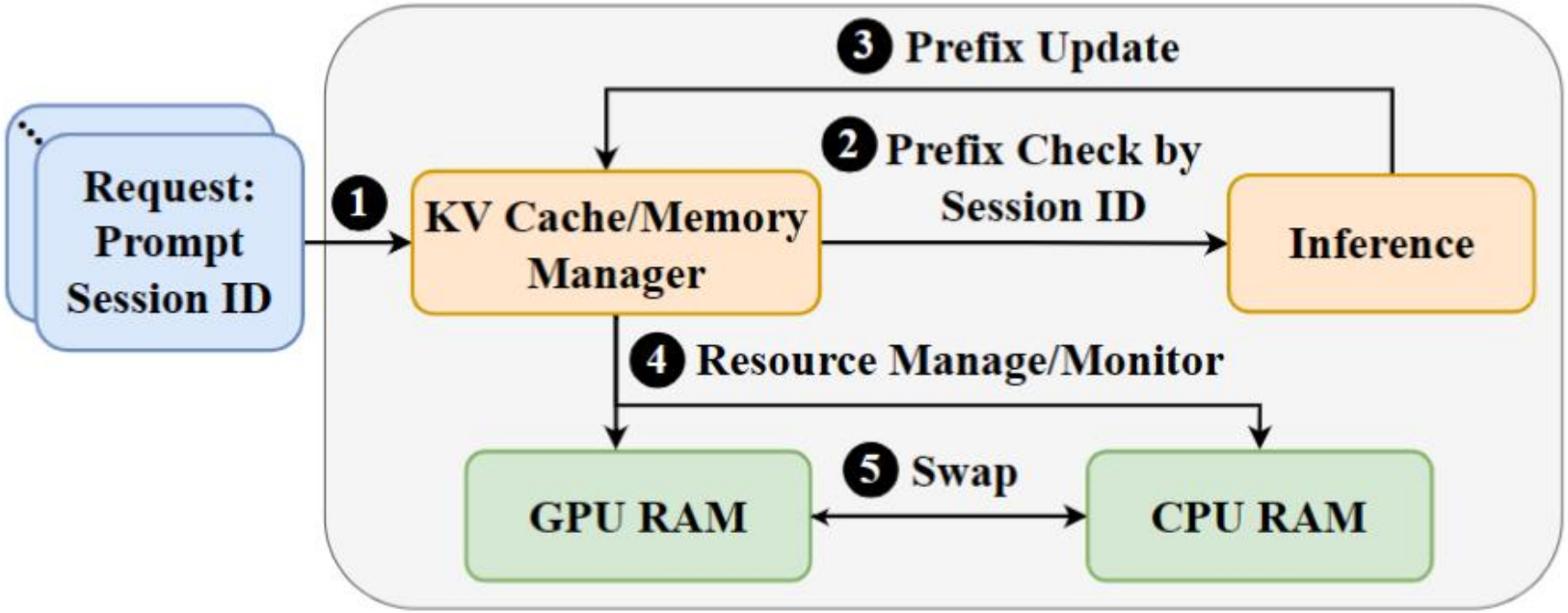
跨请求KV缓存复用的存储问题

- 当前LLM推理引擎：完成推理请求会立即释放对应的KV缓存
- 跨请求KV缓存复用：完成推理请求保留对应的KV缓存
- GPU显存没有足够的空间放置KV缓存
- 利用更大的CPU内存来放置KV缓存
- 250 tokens for Llama3-8B: A10 42ms vs. PCIE 4.8ms



KVsail设计

- 基于SessionID的会话管理机制
- KV缓存动态卸载到CPU内存



会话管理机制

- 会话信息：（SessionID，token list，KV cache，timestamp）
- 请求到来：通过SessionID查找并比对token list，将公共KV cache传入推理引擎
- 请求结束：更新会话信息
- 会话结束：释放会话信息

动态卸载机制

- 监控GPU显存利用，显存不足时，根据时间戳将KV缓存交换到CPU内存中
- 监控CPU内存，当内存不足时，根据时间戳释放KV缓存
- 当推理请求的会话信息不在GPU显存上，会从CPU内存中查找，如果依旧没有命中，那么会在GPU上重新计算

实验配置

- 机器配置
  - NVIDIA A10 GPU (24GB)
  - 256 GB CPU memory
- 实验场景
  - 多轮对话: ChatGLM3-6B
  - 代码补全: StarCoderBase-7B
- 对比方法
  - TensorRT-LLM
  - vLLM
  - ChunkedPrefill
  - PrefixCache

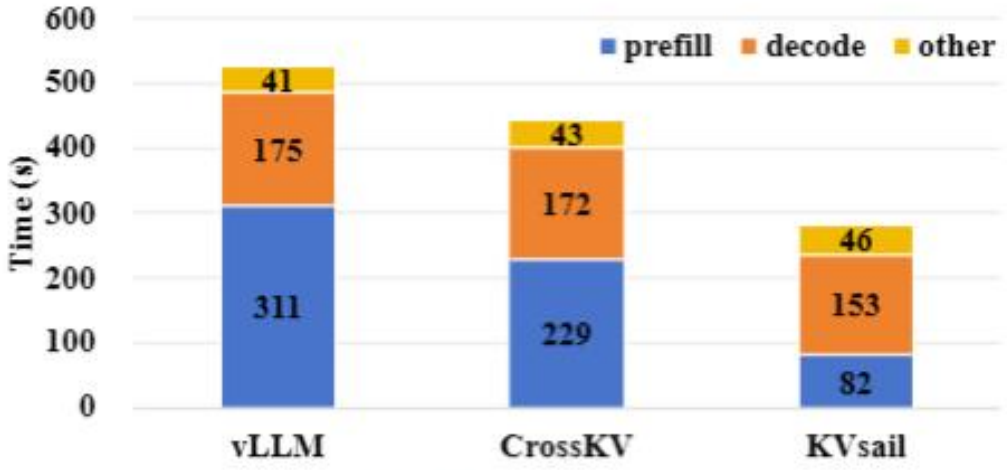
总体评估结果

Table 1: Overall results of TRTLLM, vLLM, PrefixCache, ChunkedPrefill and KVsail across multi-round chat and code completions.

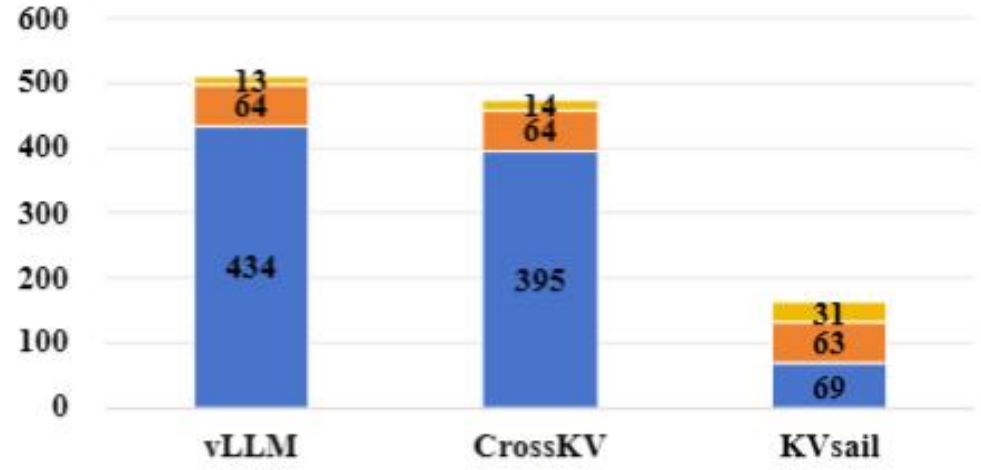
Scene	Scheme	Through. (toks/s)	Through. (reqs/s)	Latency (s)	TFT (ms)
Chat	TRTLLM	2415	2.30	127	59870
	vLLM	3481	3.32	42	2470
	ChunkedPrefill	3688	3.52	60	20672
	PrefixCache	4764	4.55	29	**1403**
	KVsail	**6543**	**6.24**	**22**	1712
Code	TRTLLM	3287	4.46	117	49636
	vLLM	3688	5.01	55	11191
	ChunkedPrefill	3571	4.85	93	45128
	PrefixCache	3990	5.42	54	11223
	KVsail	**11602**	**15.75**	**17**	**4137**

消融实验

- 对比方法
  - ➢vLLM： KVsail去掉会话管理和动态卸载机制
  - ➢CrossKV： KVsail去掉动态卸载机制
- 减少的时间主要来源于prefill阶段



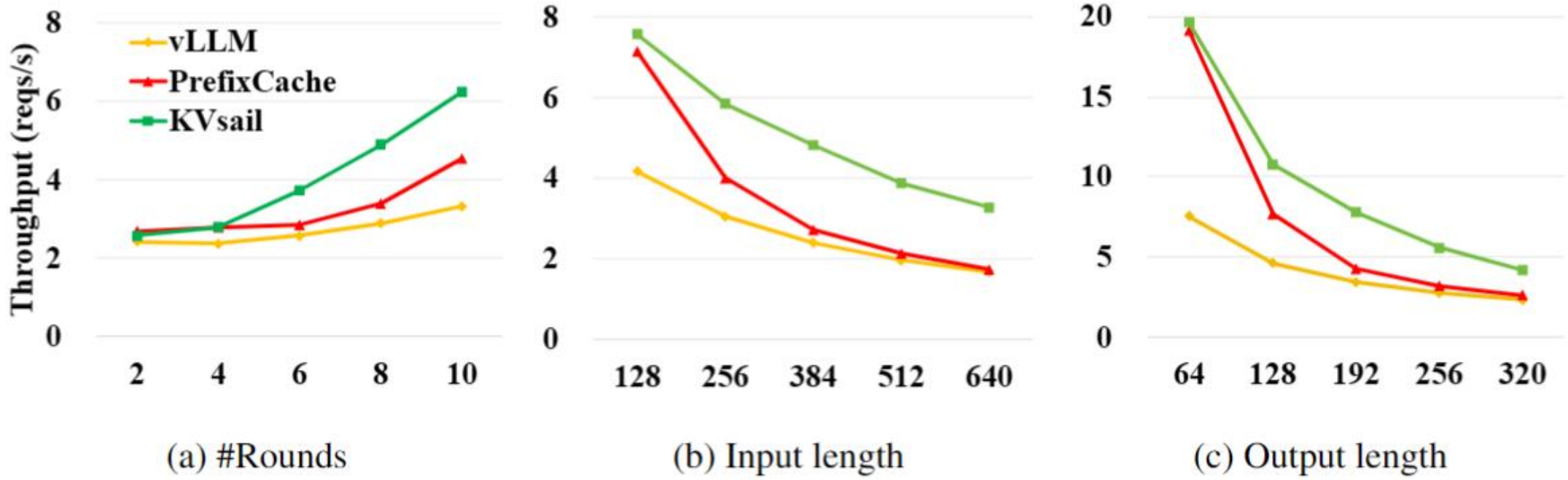
(a) Multi-round chat



(b) Code completion

参数研究

- 随着对话轮数提升，KVsail优势在变大
- 随着输入/输出长度的增加，PrefixCache相比vLLM的优势在逐渐缩小



总结

- 用户在与LLM的多轮交互过程中，存在重复的前缀信息
- 通过复用LLM推理中使用的KV缓存，可以减少冗余的计算
  - 请求内
  - 请求间
- 提出了跨请求的KV缓存管理机制
  - 基于SessionID的会话管理
  - KV缓存动态卸载
- KVsail的性能超越了SOTA方法
  - 37%/190%的吞吐量提升
  - 24%/68%的延迟降低