PR #19746 完整报告

sgl-project/sglang

[P/D disagg] - support decode side radix cache

合并时间：2026-05-01 21:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/19746

执行摘要

- 一句话：PD 拆分 decode 端支持 radix 缓存，减少 KV 传输
- 推荐动作：建议所有从事 PD 拆分和 KV 缓存优化的工程师仔细阅读该 PR。关键设计决策包括：（1）lock_ref 平衡策略及其失败路径处理；（2）将 decode prefix handoff 迁入 kv_manager state 以避免调度线程遍历 transfer_infos；（3）页面对齐和单调游标维护以防协议错误。该 PR 为后续扩展（Mooncake 支持、批处理 eviction、retraction 支持）奠定了良好基础。

功能与动机

在 PD 拆分中，decode 节点每轮都要接收完整的 KV 前缀，导致带宽浪费和 TTFT 居高不下。通过在 decode 侧启用 radix 缓存，共享前缀可被本地缓存，decode 只请求新增 KV 页面，显著降低传输延迟和内存占用。

实现拆解

1. decode 调度器前缀匹配与锁定：在 pop_preallocated 中调用新增的 _match_prefix_and_lock，通过 match_prefix_for_req 匹配 decode 侧 radix 树并锁定匹配节点，防止驱逐。获取 prefix_len 后用于预分配预算计算，并传给 _pre_alloc 初始化 extend 输入。
2. prefill 端增量传输：修改 pop_bootstrapped，通过 KV 管理器获取 decode_prefix_len，计算待发送页数（总页数 - 前缀页数）。在 send_kv_chunk 中保持发送游标单调递增，当游标超过分块边界时发送空块。add_transfer_request 在 kv_indices 为空时跳过 RDMA 传输，但辅助数据仍然发送。TransferInfo 新增 decode_prefix_len 字段，修正 is_dummy 方法以区分全命中与 CP 非权威 rank。
3. 通用工具函数：在 common.py 新增 kv_to_page_indices、kv_to_page_num、page_align_floor，将原本在 disaggregation/utils.py 中的函数通用化。新增 maybe_cache_unfinished_req 包装 cache_unfinished_req，支持 skip_radix_cache_insert 标记。调整 release_kv_cache 在 cache finished 时使用该标记。
4. CLI 与兼容性：在 server_args.py 新增 --disaggregation-decode-enable-radix-cache 布尔标志。decode 模式下启用该标志会禁止 disable_radix_cache，当前仅允许与 nixl 后端组合，并对 speculative decoding、Mamba/SSM/SWA 模型添加断言阻止。

5. 测试覆盖：新增单元测试 test_decode_radix_lock_ref.py 验证四种 transfer 场景下 lock_ref 平衡；新增分布式集成测试 test_disaggregation_decode_radix_cache.py 在 8-GPU H20 CI 上多轮缓存命中测试；更新 CI 脚本安装 NIXL。

关键文件：

- python/sglang/srt/disaggregation/decode.py（模块 解码调度；类别 source；类型 core-logic；符号 _match_prefix_and_lock, _pre_alloc, _required_alloc_tokens）：核心变更：添加 _prefix_match_and_lock、修改 pop_preallocated 和 _pre_alloc 以集成 decode 侧 radix 缓存。
- python/sglang/srt/mem_cache/common.py（模块 缓存工具；类别 source；类型 dependency-wiring；符号 kv_to_page_indices, kv_to_page_num, page_align_floor, maybe_cache_unfinished_req）：新增通用工具函数（kv_to_page_indices、page_align_floor 等）并调整 release_kv_cache 以支持 skip_radix_cache_insert 标记。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 传输协议；类别 source；类型 core-logic；符号 pop_decode_prefix_len, should_send_kv_chunk, TransferInfo）：传输协议扩展：TransferInfo 添加 decode_prefix_len，add_transfer_request 支持空 KV 传输跳过，修正 is_dummy 方法。
- python/sglang/srt/server_args.py（模块 配置管理；类别 source；类型 core-logic；符号 _handle_pd_disaggregation）：CLI 参数和兼容性验证：新增 --disaggregation-decode-enable-radix-cache，禁止与不支持的 backends 和模型特性组合。
- python/sglang/srt/managers/schedule_policy.py（模块 调度策略；类别 source；类型 refactor；符号 match_prefix_for_req）：提取 match_prefix_for_req 函数，供 decode 调度器和普通调度器共用。
- test/registered/unit/mem_cache/test_decode_radix_lock_ref.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestDecodeLockRefScenarios, _make_cache_with_pools, MockReq）：单元测试覆盖四种 transfer 锁平衡场景，验证 inc/dec_lock_ref 无泄漏。

关键符号：_match_prefix_and_lock, _pre_alloc, _required_alloc_tokens, match_prefix_for_req, kv_to_page_indices, kv_to_page_num, page_align_floor, maybe_cache_unfinished_req, TransferInfo.is_dummy, TransferInfo.from_zmq, pop_decode_prefix_len, should_send_kv_chunk

关键源码片段

python/sglang/srt/mem_cache/common.py

新增通用工具函数（kv_to_page_indices、page_align_floor 等）并调整 release_kv_cache 以支持 skip_radix_cache_insert 标记。

```
def kv_to_page_indices(kv_indices: np.ndarray, page_size: int):
    # 页面除最后一块外保证是完整的
    if page_size == 1:
        return kv_indices
    return kv_indices[::page_size] // page_size
```

```
def kv_to_page_num(num_kv_indices: int, page_size: int):
    return (num_kv_indices + page_size - 1) // page_size


def page_align_floor(length: int, page_size: int) -> int:
    return (length // page_size) * page_size


def maybe_cache_unfinished_req(req: Req, tree_cache: BasePrefixCache, **kwargs):
    # 当标记为跳过 radix 缓存插入时（例如 decode radix cache 场景），直接返回
    if getattr(req, "skip_radix_cache_insert", False):
        return
    tree_cache.cache_unfinished_req(req, **kwargs)
```

python/sglang/srt/disaggregation/nixl/conn.py

传输协议扩展：TransferInfo 添加 decode_prefix_len，add_transfer_request 支持空 KV 传输跳过，修正 is_dummy 方法。

```
@dataclasses.dataclass
class TransferInfo:
    """传输信息，包含从 decode 发送到 prefill 的索引和参数。"""
    room: int
    endpoint: str
    dst_port: int
    agent_name: str
    dst_kv_indices: npt.NDArray[np.int32]
    dst_aux_index: int
    required_dst_info_num: int
    dst_state_indices: List[int]
    decode_prefix_len: Optional[int] = None # 新增：decode 侧已缓存的前缀长度

    def is_dummy(self):
        # CP 非权威 rank 的传输为 dummy；
        # 但当 kv_indices 为空且 decode_prefix_len > 0（全命中）时不是 dummy
        if self.dst_kv_indices.size == 0 and self.decode_prefix_len:
            return False
        return self.dst_kv_indices.size == 0

    @classmethod
    def from_zmq(cls, msg: List[bytes]):
        # 解析 state_indices
        dst_state_indices = []
        if len(msg) > 7 and msg[7] != b"":
            dst_state_indices = list(np.frombuffer(msg[7], dtype=np.int32))
        return cls(
            room=int(msg[0].decode("ascii")),
            endpoint=msg[1].decode("ascii"),
            dst_port=int(msg[2].decode("ascii")),
            agent_name=msg[3].decode("ascii"),
```

```
            dst_kv_indices=np.frombuffer(msg[4], dtype=np.int32),
            dst_aux_index=int(msg[5].decode("ascii")),
            required_dst_info_num=int(msg[6].decode("ascii")),
            dst_state_indices=dst_state_indices,
            decode_prefix_len=(
                int(msg[8].decode("ascii")) if len(msg) > 8 and msg[8] != b"" else None
            ),
        )
```

评论区精华

- Transfer 失败路径 lock_ref 平衡（@yudian0504）：总结四种场景的 lock_ref 变化，要求验证所有路径。作者通过单元测试覆盖成功和失败路径，确认无泄漏。（已解决）
- 非 NIXL 后端兼容性（@ShangmingCai）：询问 Mooncake 等其他后端是否支持。作者暂时仅允许 NIXL 并添加 ERROR，后续 PR 添加支持。（已解决）
- Retraction 与 radix 缓存交互（@cctry）：指出 retraction 需要正确处理前缀匹配，目前未实现。作者添加 TODO，计划后续 PR 添加。（待处理）
- Warmup 垃圾 KV 处理（@cctry）：warmup 请求使用伪造 bootstrap 地址，产生的 KV 应该被丢弃。作者在 warmup 结束后 flush radix 缓存解决。（已解决）
- Transfer 失败路径 lock_ref 平衡 (correctness): 作者通过新增单元测试 test_decode_radix_lock_ref.py 覆盖四种场景，确认 inc/dec_lock_ref 平衡且无泄漏。
- 非 NIXL 后端兼容性 (design): 当前仅允许 NIXL 后端；Mooncake 支持将在后续 PR 中添加。
- Retraction 与 radix 缓存交互 (correctness): 作者确认未实现，并添加 TODO 注释，计划后续 PR 处理。
- Warmup 垃圾 KV 处理 (correctness): 作者通过 warmup 结束后执行 flush_cache 清除所有 garbage KV（commit a6b6e8b）。

风险与影响

- 风险：
  - lock_ref 泄漏风险：如果 transfer 失败后 lock_ref 未正确递减，将导致缓存行永久锁定，最终耗尽内存。已通过单元测试覆盖四种场景，但实际部署中仍可能遇到未预见的失败路径。
  - 页对齐不一致：decode 和 prefill 对前缀长度进行页对齐时若算法不同，可能导致传输内容错位。代码在 pop_preallocated 中使用 page_align_floor 对齐，并在 kv_to_page_indices 中处理。
  - 非 NIXL 后端：当前仅支持 NIXL，若其他后端被启用，将因 decode_prefix_len 参数缺失而崩溃。已在 server_args.py 中禁止其他后端，但后续添加 Mooncake 时需额外工作。
  - DP attention 命中率：在 DP 模式下，如果同一对话被路由到不同 rank，缓存命中率将大幅下降。需要外部路由器支持 prefix-aware 路由。
  - eviction 未批处理：当前每个请求独立执行 eviction，可能在高负载下成为性能瓶颈，计划后续批处理优化。

- 影响：
  - 用户影响：新增 CLI 选项，用户可在 PD 拆分 *decode* 节点上启用 radix 缓存，获得显著的 TTFT 降低和吞吐提升（实测 TTFT P50 降低 8.1 倍，吞吐提升 1.32 倍），但 ITL 可能因 batch 变大而退化。
  - 系统影响：*decode* 节点内存利用率提高（更多请求可同时运行），但增加的 batch size 可能增加解码延迟。对 prefill 节点影响有限，仅减少发送的 KV 页面数。
  - 团队影响：核心团队需为 *Mooncake* 等后端适配同样功能；调度团队需考虑 *prefix-aware* 路由支持；测试团队需维护新增的分布式测试。
  - 风险标记：核心路径变更，lock_ref 泄漏风险，非 NIXL 后端未支持，*eviction* 未批处理

关联脉络

- 暂无明显关联 PR