PR #21125 完整报告

sgl-project/sglang

[HiCache] feat: add draft KV cache backing for L2/L3

合并时间：2026-04-29 14:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21125

执行摘要

- 一句话：在 HiCache 中同步 draft KV 缓存修复 accept length 退化
- 推荐动作：本 PR 修复了一个关键的 Spec+HiCache 协同 bug，设计合理（piggyback 式同步），且提供了完整的 benchmark 数据验证。此外，_get_draft_kv_pool 的抽取体现了良好的重构意识。建议阅读 _maybe_register_hicache_draft 和 start_writing 中的同步逻辑，理解如何在现有框架下优雅地加入辅助缓存池。

功能与动机

关联 Issue #16964 报告了 DeepSeek 模型在 B200 上启用 Spec + HiCache 后 accept length 严重退化。根本原因是 draft 与 target 共享 req_to_token_pool 但 KV cache pool 分离，target load_back 后 shared index 更新了但 draft 的 KV cache 未同步。通过成对 offload/load_back 目标与 draft 的 KV cache 来彻底修复。

实现拆解

1. 在调度器中注册 draft KV 池：新增 _get_draft_kv_pool() 方法从 draft worker 获取 token_to_kv_pool，并在 _maybe_register_hicache_draft() 中根据 pool 类型创建对应的 host pool，然后通过 set_draft_kv_pool 注册到 cache_controller。
2. 在 cache_controller 中同步 draft 操作：start_writing() 和 start_loading() 在操作 target KV 后备注同步调用 draft host/device pool 的备份与恢复；_page_transfer() 和 _page_get 对 L3 读写时带上 draft 前缀键（d:）实现 best-effort 同步。
3. 新增端到端测试：test_hicache_spec_file_storage.py 覆盖 EAGLE3 模型下 HiCache file 存储的 offload/load_back 场景，验证 accept length 不低于阈值。

关键文件：

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _get_draft_kv_pool, _maybe_register_hicache_draft）：新增 _get_draft_kv_pool 和 _maybe_register_hicache_draft 方法，在初始化阶段将 draft KV 池注册到 HiCache 控制器，是入口核心。
- python/sglang/srt/managers/cache_controller.py（模块 缓存控制器；类别 source；类型 entrypoint；符号 set_draft_kv_pool, _draft_page_set, _draft_page_get）：核心同步逻辑所在：在 start_writing、start_loading、_page_transfer 等方法中插入 draft KV 的备份与恢复操作。

- test/registered/hicache/test_hicache_spec_file_storage.py（模块 测试；类别 test；类型 test-coverage；符号 TestHiCacheSpecFileStorage, setUpClass, _launch_server, _stop_server）：新增 E2E 测试，验证 HiCache + EAGLE3 在 file 存储后端下 accept length 不退化。

关键符号：set_draft_kv_pool, _draft_page_set, _draft_page_get, _get_draft_kv_pool, _maybe_register_hicache_draft

关键源码片段

python/sglang/srt/managers/scheduler.py

新增 _get_draft_kv_pool 和 _maybe_register_hicache_draft 方法，在初始化阶段将 draft KV 池注册到 HiCache 控制器，是入口核心。

```
# python/sglang/srt/managers/scheduler.py

def _get_draft_kv_pool(self):
    """从 draft worker 获取 token_to_kv_pool，支持多层 Eagle 只取第一层。"""
    if self.draft_worker is None or self.spec_algorithm.is_ngram():
        return None, None
    if self.spec_algorithm.supports_spec_v2() and self.enable_overlap:
        if self.server_args.enable_multi_layer_eagle:
            # 多层 Eagle 只同步第一层 draft runner
            draft_runner = self.draft_worker.draft_worker.draft_runner_list[0]
        else:
            draft_runner = self.draft_worker.draft_worker.draft_runner
        return draft_runner.token_to_kv_pool, draft_runner.model_config
    return (
        self.draft_worker.model_runner.token_to_kv_pool,
        self.draft_worker.model_config,
    )

def _maybe_register_hicache_draft(self) -> None:
    """如果启用了 HiCache 并且有 draft worker，则注册 draft KV pool。"""
    if not self.enable_hierarchical_cache:
        return
    draft_kv_pool, _ = self._get_draft_kv_pool()
    if draft_kv_pool is None:
        return
    # 根据 pool 类型创建对应的 host pool，确保 slot 数量与 target 一致
    from sglang.srt.mem_cache.memory_pool import HybridLinearKVPool, MHATokenToKVPool,
    MLATokenToKVPool
    from sglang.srt.mem_cache.memory_pool_host import MHATokenToKVPoolHost,
    MLATokenToKVPoolHost
    pool = draft_kv_pool
    if isinstance(pool, HybridLinearKVPool):
        pool = pool.full_kv_pool
    primary = self.tree_cache.cache_controller.mem_pool_host
    kw = dict(
```

```
        host_to_device_ratio=primary.size / pool.size,
        host_size=0,
        page_size=self.page_size,
        layout=self.server_args.hicache_mem_layout,
    )
    if isinstance(pool, MHATokenToKVPool):
        draft_host_pool = MHATokenToKVPoolHost(pool, **kw)
    elif isinstance(pool, MLATokenToKVPool):
        draft_host_pool = MLATokenToKVPoolHost(pool, **kw)
    else:
        logger.warning("Draft pool type %s not supported for HiCache, skipping.", type(pool).__
        name__)
        return
    self.tree_cache.cache_controller.set_draft_kv_pool(pool, draft_host_pool)
```

python/sglang/srt/managers/cache_controller.py

核心同步逻辑所在：在 start_writing、start_loading、_page_transfer 等方法中插入 draft KV 的备份与恢复操作。

```
# python/sglang/srt/managers/cache_controller.py

def set_draft_kv_pool(self, draft_device_pool, draft_host_pool) -> None:
    """注册 draft KV 池，之后所有 L2/L3 操作都会同步操作 draft。"""
    self.has_draft = True
    self.mem_pool_device_draft = draft_device_pool
    self.mem_pool_host_draft = draft_host_pool
    logger.info(
        "HiCache draft KV registered: %s (host %d slots)",
        type(draft_device_pool).__name__,
        draft_host_pool.size,
    )

def start_writing(self) -> None:
    # ... 原有逻辑 ...
    with device_module.stream(self.write_stream):
        # 先备份 target KV
        self.mem_pool_host.backup_from_device_all_layer(
            self.mem_pool_device, host_indices, device_indices, self.io_backend
        )
        # 如果注册了 draft，则同步备份 draft KV
        if self.has_draft:
            self.mem_pool_host_draft.backup_from_device_all_layer(
                self.mem_pool_device_draft,
                host_indices,
                device_indices,
                self.io_backend,
            )
    # ...
```

```
def start_loading(self) -> int:
    # ... 逐层加载 target KV ...
    for i in range(self.layer_num):
        self.mem_pool_host.load_to_device_per_layer(
            self.mem_pool_device, host_indices, device_indices, i, self.io_backend
        )
        # 如果 draft 层数足够则同步加载
        if self.has_draft and i < self.mem_pool_host_draft.layer_num:
            self.mem_pool_host_draft.load_to_device_per_layer(
                self.mem_pool_device_draft,
                host_indices,
                device_indices,
                i,
                self.io_backend,
            )
```

test/registered/hicache/test_hicache_spec_file_storage.py

新增 E2E 测试，验证 HiCache + EAGLE3 在 file 存储后端下 accept length 不退化。

```
# test/registered/hicache/test_hicache_spec_file_storage.py

import json, os, shutil, tempfile, time, unittest
from typing import Dict, List
import psutil, requests
from sglang.benchmark.utils import get_tokenizer
from sglang.srt.utils import is_hip, kill_process_tree
from sglang.test.ci.ci_register import register_cuda_ci
from sglang.test.test_utils import (
    DEFAULT_DRAFT_MODEL_EAGLE3, DEFAULT_TARGET_MODEL_EAGLE3,
    DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH, DEFAULT_URL_FOR_TEST,
    CustomTestCase, find_available_port, popen_launch_server,
)
from sglang.utils import wait_for_http_ready

register_cuda_ci(est_time=600, suite="stage-b-test-1-gpu-large")

@unittest.skipIf(is_hip(), "HiCache + EAGLE3 file-storage loadback e2e is CUDA-only.")
class TestHiCacheSpecFileStorage(CustomTestCase):
    model = DEFAULT_TARGET_MODEL_EAGLE3
    draft_model = DEFAULT_DRAFT_MODEL_EAGLE3
    input_token_len = 1024
    max_new_tokens = 200
    page_size = 64
    min_expected_accept_length = 7.0
    min_second_to_first_accept_ratio = 0.9
    storage_wait_timeout = 30
    first_measure_new_tokens = 128

    @classmethod
```

```
    def setUpClass(cls):
        cls.temp_dir = tempfile.mkdtemp()
        default_port = int(DEFAULT_URL_FOR_TEST.rsplit(":", 1)[1])
        cls.base_url = f"http://127.0.0.1:{find_available_port(default_port)}"
        cls.tokenizer = get_tokenizer(cls.model)
        cls.prompt_input_ids = cls._build_long_repetitive_prompt_ids(cls.tokenizer, cls.input_token_
        len)
        extra_config = {"hicache_storage_pass_prefix_keys": True}
        cls.other_args = [
            "--enable-hierarchical-cache", "--enable-cache-report",
            "--mem-fraction-static", "0.3", "--hicache-ratio", "1.5",
            "--disable-cuda-graph", "--page-size", str(cls.page_size),
            "--hicache-storage-backend", "file",
            "--hicache-storage-prefetch-policy", "wait_complete",
            "--hicache-storage-backend-extra-config", json.dumps(extra_config),
            "--speculative-algorithm", "EAGLE3",
            "--speculative-draft-model-path", cls.draft_model,
            "--speculative-num-steps", "7", "--speculative-eagle-topk", "1",
            "--speculative-num-draft-tokens", "8", "--dtype", "float16",
        ]
        cls.env = {
            **os.environ,
            "SGLANG_ALLOW_OVERWRITE_LONGER_CONTEXT_LEN": "1",
            "SGLANG_HICACHE_FILE_BACKEND_STORAGE_DIR": cls.temp_dir,
        }
        cls.process = None
        cls._launch_server()
    # ...（其余测试方法：_launch_server, test_accept_length, _count_file_storage_pages 等）
```

评论区精华

- 重复逻辑抽取建议：代码审阅者 gemini-code-assist[bot] 指出 _maybe_register_hicache_draft 中定位 draft KV pool 的逻辑与 init_disaggregation 重复，建议抽取为 _get_draft_kv_pool 公用方法。该建议已被采纳，最终代码中已提取该方法并在两处调用。
- 前缀冲突疑虑：stmatengss 对 d: 前缀容易冲突表示担忧。alphabetc1 回应 target 的 key 是 SHA256 hex，只包含 0-9a-f，不含冒号，因此不会冲突。讨论已解决。
- EAGLE3 兼容性：ovowei 询问能否用于 EAGLE3，作者确认可以但对多层 Eagle 效果可能不完美。
  - 抽取 _get_draft_kv_pool 避免重复逻辑 (design): 作者采纳建议，最终代码中已提取 _get_draft_kv_pool 方法并在两处调用。
  - draft 前缀键 "d:" 冲突风险 (correctness): alphabetc1 解释 target key 是 SHA256 hex，不包含冒号，因此不会冲突。

风险与影响

- 风险：

1. 性能开销：每次 target KV 的写 / 加载都会额外复制一次 draft KV，增加 GPU 到 host 的带宽消耗，但 benchmark 显示整体延迟下降，表明收益大于开销。
2. 内存占用：为 draft 额外分配 host pool（与 target 等大），在内存受限场景可能加剧压力。
3. 兼容性限制：目前仅支持 MHATokenToKVPool 和 MLATokenToKVPool 类型，其他 pool 类型会被跳过。对多层 Eagle（如 EAGLE3）仅同步第一层 draft。
4. 测试覆盖：端到端测试仅覆盖 EAGLE3 + file 后端，其他算法（如 Medusa）或存储后端未验证。 - 影响：用户层面：修复了 Spec+HiCache 启用时性能倒挂的 bug，让 HiCache 在长序列命中场景中真正发挥加速作用。系统层面：增加了约 68 行的核心调度逻辑和 81 行的缓存控制同步逻辑，每次操作额外多一次 draft 数据传输。团队层面：新增一个 E2E 测试文件，CI 时长增加约 600 秒，但保证质量。整体影响可控，为正向改进。 - 风险标记：核心路径变更，内存占用增加，兼容性限制，仅 EAGLE3 验证

关联脉络

- 暂无明显关联 PR