PR #20977 完整报告

sgl-project/sglang

[HiCache] Add CP support for HiCache

合并时间：2026-04-10 17:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/20977

执行摘要

- 一句话：为 HiCache 添加注意力上下文并行（CP）支持，确保缓存键正确生成。
- 推荐动作：建议技术管理者和工程师精读此 PR，特别关注 mooncake_store.py 中的缓存键生成逻辑和设计讨论。这有助于理解在异构并行设置下如何维护缓存一致性。

功能与动机

根据 PR 描述，此变更主要为支持 Qwen3 CP + Hicache，因为 MLA 模型在使用 HiCache 时会复用 CP rank 0 的数据，因此无需区分键，但需要将 CP 信息整合到缓存系统中以确保正确性。

实现拆解

1. 添加 CP 参数到配置类：在 python/sglang/srt/mem_cache/cache_init_params.py 的 CacheInitParams 类和 python/sglang/srt/mem_cache/hicache_storage.py 的 HiCacheStorageConfig 类中添加 attn_cp_rank 和 attn_cp_size 字段，为参数传递提供基础。
2. 传递参数到控制器和缓存：在 python/sglang/srt/managers/scheduler.py 的 init_cache_with_memory_pool 方法中，将 attn_cp_rank 和 attn_cp_size 传递给 CacheInitParams；在 python/sglang/srt/mem_cache/hiradix_cache.py 的 __init__ 方法中接收并存储这些参数，并传递给 HiCacheController。
3. 调整缓存键生成逻辑：在 python/sglang/srt/mem_cache/storage/mooncake_store/mooncake_store.py 的 __init__ 方法中，添加 attn_cp_rank 和 attn_cp_size，并修改 mha_suffix 和 mla_suffix 的生成，当启用 PP 或 CP 时包含 CP rank。
4. 更新监控指标标签：在 python/sglang/srt/mem_cache/hiradix_cache.py 的 _apply_storage_runtime_config 方法中，将 attn_cp_rank 和 attn_cp_size 添加到 metrics 标签，以便监控。

关键文件：

- python/sglang/srt/mem_cache/storage/mooncake_store/mooncake_store.py（模块 缓存存储；类别 source；类型 core-logic；符号 init）：这是缓存存储的核心逻辑文件，直接处理缓存键生成，变更最多且涉及控制流调整。
- python/sglang/srt/managers/cache_controller.py（模块 缓存控制；类别 source；类型 entrypoint；符号 init, _generate_storage_config）：作为 HiCache 的控制器入口，负责

管理缓存存储，变更添加了 CP 参数到初始化和配置生成。

- python/sglang/srt/mem_cache/hiradix_cache.py（模块 缓存树；类别 source；类型 core-logic；符号 init, _apply_storage_runtime_config）：作为 HiCache 的核心逻辑层，负责初始化缓存控制器并传递 CP 参数，同时更新监控指标。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 init_cache_with_memory_pool）：作为调度器，负责初始化缓存参数，变更将 CP 信息传递给 CacheInitParams。
- python/sglang/srt/mem_cache/cache_init_params.py（模块 缓存参数；类别 source；类型 data-contract；符号 CacheInitParams）：定义了缓存初始化参数的基类，变更添加 CP 字段以支持参数传递链条。
- python/sglang/srt/mem_cache/hicache_storage.py（模块 存储配置；类别 source；类型 data-contract；符号 HiCacheStorageConfig）：定义了缓存存储配置的数据类，变更添加 CP 字段以扩展配置信息。

关键符号：CacheInitParams.init, HiCacheStorageConfig.init, HiCacheController.init, MooncakeStore.init, HiRadixCache.init, Scheduler.init_cache_with_memory_pool

关键源码片段

python/sglang/srt/mem_cache/storage/mooncake_store/mooncake_store.py

这是缓存存储的核心逻辑文件，直接处理缓存键生成，变更最多且涉及控制流调整。

```
def __init__(self, storage_config=None, ...):
    # ... 其他初始化代码 ...
    if storage_config is not None:
        self.is_mla_backend = storage_config.is_mla_model
        self.local_rank = storage_config.tp_rank
        self.pp_rank = storage_config.pp_rank
        self.pp_size = storage_config.pp_size
        self.attn_cp_rank = storage_config.attn_cp_rank # 新增：注意力 CP rank
        self.attn_cp_size = storage_config.attn_cp_size # 新增：注意力 CP size
        self.enable_storage_metrics = storage_config.enable_storage_metrics
    else:
        self.is_mla_backend = False
        self.local_rank = 0
        self.pp_rank = 0
        self.pp_size = 1
        self.attn_cp_rank = 0 # 默认值
        self.attn_cp_size = 1 # 默认值

    self.enable_pp = self.pp_size > 1
    self.enable_cp = self.attn_cp_size > 1 # 新增：检查 CP 是否启用
    if self.enable_pp or self.enable_cp: # 逻辑调整：PP 或 CP 启用时包含所有秩
        self.mha_suffix = f"{self.local_rank}_{self.pp_rank}_{self.attn_cp_rank}"
        self.mla_suffix = f"{self.pp_rank}_{self.attn_cp_rank}"
    else:
        self.mha_suffix = f"{self.local_rank}"
```

```
        self.mla_suffix = ""
    # ... 后续代码，例如处理 should_split_heads 等 ...
```

评论区精华

reviewer stmatengss 提出是否应分别检查 enable_pp 和 enable_cp，或者当仅启用 PP 时 attn_cp_rank 应为默认值 0。作者 ShangmingCai 回复表示，如果支持异构设置，可能应让 suffix 始终包含所有秩，即使为 0，并考虑将 pp size 和 cp size 也加入 suffix。这揭示了设计决策中的权衡。

- 缓存键生成中的 PP 和 CP 检查 (design): 作者 ShangmingCai 回复表示，如果支持异构设置，可能应该让 suffix 始终包含所有秩，即使为 0，并考虑将 pp size 和 cp size 也加入 suffix。

风险与影响

- 风险：技术风险包括：
  1. 缓存一致性风险：如果 CP 模式下缓存键生成不正确，可能导致数据不一致或错误缓存命中。
  2. 兼容性问题：新增参数可能影响现有配置，尤其是在未启用 CP 时默认值的处理。
  3. 缺少测试覆盖：从变更文件看，没有直接对应的测试文件更新，可能缺乏对 CP 支持的全面验证。
  4. 与 PR #20460 的协调：讨论中提到需要同步缓存状态，可能存在未解决的依赖。 - 影响：对用户影响：使用 CP 并启用 HiCache 的用户（如 Qwen3 模型）将受益于更好的缓存支持，可能提升推理性能。对系统影响：扩展了 HiCache 的功能范围，使其更适应并行设置。对团队影响：需要关注与相关 PR（如 #20460）的集成，确保缓存状态同步问题得到解决。 - 风险标记：缓存一致性风险，兼容性问题，缺少测试覆盖

关联脉络

- PR #20460 未知：在 Issue 评论中被提及，涉及缓存状态同步，与本 PR 的 CP 支持相关，可能需要协调以完善功能。