

PR #49858 完整报告

vllm-project/vllm

[KV Offload] Make compact secondary identity TP-independent

合并时间: 2026-07-28 12:10

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/49858>

执行摘要

- 一句话: 使紧凑 MLA 行在不同 TP 大小下共享二级缓存路径
- 推荐动作: 建议精读。此 PR 设计将 TP 无关 identity 的职责明确置于 FileMapper, 通过单一 discriminator 字段隔离不同 layout, 实现简洁清晰。测试覆盖充分, 特别是跨 TP 字节级 round-trip 测试。合并后建议监控二级缓存命中率变化。

功能与动机

完成 #47929 中规划的 secondary-tier 步骤。#48906 使单节点 TP-only 的 MLA primary row 紧凑化, 但 secondary-tier identity 仍包含 TP 配置 (FS/OBJ 路径关联 TP 大小, P2P 兼容性指纹也基于相同字段)。本 PR 通过 replicated_layout 解耦 TP 身份, 使紧凑行能在不同 TP 大小间共享二级缓存。

实现拆解

1. FileMapper 核心逻辑扩展 (vllm/v1/kv_offload/file_mapper.py) - `__init__` 新增 `replicated_layout: bool = False` 参数, 当为 True 时将 "replicated_layout": True 写入 fields (不写 False 以保持历史 hash 不变)。- `from_offloading_spec` 中将 `parallel_agnostic` 的计算条件从 `parallel_agnostic and parallel.is_parallelism_agnostic` 扩展为 `parallel_agnostic and (parallel.is_parallelism_agnostic or config.replicated_layout)`, 使 replicated 行也能进入 parallel-agnostic 路径。同时传递 `replicated_layout=(parallel_agnostic and config.replicated_layout)` 给构造函数。
2. 测试辅助函数参数化 (tests/v1/kv_offload/test_file_mapper.py、test_fs_tier.py、test_obj_tier.py) - `make_mapper_from_offloading_spec`、`_make_offloading_spec`、`_make_offloading_config` 新增 `tp_size`、`rank`、`world_size`、`replicated_layout`、`is_parallelism_agnostic` 参数, 便于在各种并行配置下构造 FileMapper。
3. 新增 FileMapper 单元测试 (test_file_mapper.py: 5 个测试) -
`test_replicated_layout_collapses_parallel_identity`: 验证 TP=2 和 TP=4 的 replicated 行拥有相同的 base_path, 且 rank/tp_size 等折叠为 1/0。 -
`test_replicated_layout_requires_caller_opt_in`: 未设置 `parallel_agnostic=True` 时 replicated 不起作用。 - `test_non_replicated_keeps_parallel_identity`: 非 replicated 行保留原始 TP 身份。 - `test_replicated_and_parallelism_agnostic_separate_layouts`: replicated 与普通 parallel-agnostic 行具有不同 base_path。 -
`test_replicated_layout_run_config_tp_invariant`: 不同 TP 的 run_config 返回相同字典。

4. 跨 TP 端到端 FS 测试 (`test_fs_tier.py: test_fs_tier_cross_tp_round_trip`) - 使用 TP=2 写入一个 block, 再用 TP=4 读取, 验证文件路径一致、数据字节级相等。
5. OBJ identity 折叠测试 (`test_obj_tier.py: test_obj_tier_replicated_layout_collapses_mapper_identity`) - 验证 TP=2 和 TP=4 的 replicated 配置下, ObjectStore tier 的 FileMapper `base_path` 和 `run_config` 完全一致。

关键文件:

- `vllm/v1/kv_offload/file_mapper.py` (模块 文件映射器; 类别 `source`; 类型 `core-logic`; 符号 `FileMapper.init`, `FileMapper.from_offloading_spec`) : 核心实现文件, 修改 `FileMapper` 的 `__init__` 和 `from_offloading_spec` 以支持 `replicated_layout`, 是 `identity` 折叠的关键入口。
- `tests/v1/kv_offload/test_file_mapper.py` (模块 映射器测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_replicated_layout_collapses_parallel_identity`, `test_replicated_layout_requires_caller_opt_in`, `test_non_replicated_keeps_parallel_identity`, `test_replicated_and_parallelism_agnostic_separate_layouts`) : 新增 5 个测试覆盖了 `replicated_layout` 的各种组合, 确保 `identity` 折叠、`opt-in` 要求、与非 `replicated` 的隔离等。
- `tests/v1/kv_offload/tiering/test_fs_tier.py` (模块 文件系统层; 类别 `test`; 类型 `test-coverage`; 符号 `_make_offloading_spec`, `test_fs_tier_cross_tp_round_trip`) : 端到端跨 TP 字节级 round-trip 测试, 直接验证 FS tier 在 `replicated_layout` 下的正确性。
- `tests/v1/kv_offload/tiering/test_obj_tier.py` (模块 对象存储层; 类别 `test`; 类型 `test-coverage`; 符号 `_make_offloading_config`, `test_obj_tier_replicated_layout_collapses_mapper_identity`) : 验证 OBJ tier 在 `replicated_layout` 下同样折叠 `FileMapper identity`, 确保不同 TP 共享路径。

关键符号: `FileMapper.init`, `FileMapper.from_offloading_spec`

关键源码片段

`vllm/v1/kv_offload/file_mapper.py`

核心实现文件, 修改 `FileMapper` 的 `__init__` 和 `from_offloading_spec` 以支持 `replicated_layout`, 是 `identity` 折叠的关键入口。

```
class FileMapper:
    def __init__(
        self,
        root_dir: str,
        model_name: str,
        tokens_per_hash: int,
        blocks_per_file: int,
        tp_size: int,
        pp_size: int,
        pcp_size: int,
        dcp_size: int,
        rank: int,
```

```

dtype: str,
kv_cache_groups: list[dict] | None = None,
inference_engine: str = "vllm",
parallel_agnostic: bool = False,
replicated_layout: bool = False, # 新增参数
):
    # 当 parallel_agnostic 为 True 时, 折叠所有并行维度到 1 和 0
    if parallel_agnostic:
        tp_size = pp_size = pcp_size = dcp_size = 1
        rank = 0
    self.rank: int = rank
    self.fields: dict = {
        "model_name": model_name,
        "tokens_per_hash": tokens_per_hash,
        "blocks_per_file": blocks_per_file,
        "tp_size": tp_size,
        "pp_size": pp_size,
        "pcp_size": pcp_size,
        "dcp_size": dcp_size,
        "dtype": str(dtype),
        "kv_cache_groups": kv_cache_groups or [],
        "inference_engine": inference_engine,
    }
    # 非 parallel_agnostic 时写入标记, 确保旧版 hash 不变
    if not parallel_agnostic:
        self.fields["parallel_agnostic"] = False
    # 仅在 replicated_layout 为 True 时写入, 避免改变旧版 hash
    if replicated_layout:
        self.fields["replicated_layout"] = True
    self.base_path: str = self._compute_base_path(root_dir, self.fields)

    @classmethod
    def from_offloading_spec(
        cls,
        root_dir: str,
        offloading_spec: OffloadingSpec,
        blocks_per_file: int = 1,
        parallel_agnostic: bool = False,
    ) -> "FileMapper":
        config = offloading_spec.config
        parallel = config.parallel
        # replicated_layout 行也被允许进入 parallel-agnostic 路径
        return cls(
            root_dir=root_dir,
            model_name=config.model.name,
            tokens_per_hash=config.cache.tokens_per_hash,
            blocks_per_file=blocks_per_file,
            tp_size=parallel.tp_size,
            pp_size=parallel.pp_size,

```

```

pcp_size=parallel.pcp_size,
dcp_size=parallel.dcp_size,
rank=parallel.rank,
dtype=config.model.dtype,
kv_cache_groups=kv_cache_groups,
parallel_agnostic=(
    parallel_agnostic
    and (parallel.is_parallelism_agnostic or config.replicated_layout)
),
replicated_layout=(parallel_agnostic and config.replicated_layout),
)

```

tests/v1/kv_offload/test_file_mapper.py

新增 5 个测试覆盖了 replicated_layout 的各种组合，确保 identity 折叠、opt-in 要求、与非 replicated 的隔离等。

```

def test_replicated_layout_collapses_parallel_identity():
    # 共享配置: replicated_layout=True + parallel_agnostic=True
    shared = dict(
        model_name="mla-model",
        groups=((16, "mla_layer"),),
        replicated_layout=True,
        parallel_agnostic=True,
    )
    # TP=2 和 TP=4 应映射到相同路径
    tp2 = make_mapper_from_offloading_spec(tp_size=2, world_size=2, rank=1, **shared)
    tp4 = make_mapper_from_offloading_spec(tp_size=4, world_size=4, rank=3, **shared)

    assert tp2.base_path == tp4.base_path
    for fm in (tp2, tp4):
        # 所有并行维度折叠为 1, rank 为 0
        assert fm.fields["tp_size"] == 1
        assert fm.fields["pp_size"] == 1
        assert fm.fields["pcp_size"] == 1
        assert fm.fields["dcp_size"] == 1
        assert fm.rank == 0
        # parallel_agnostic 字段不存在 (因为已折叠), 但 replicated_layout 保留
        assert "parallel_agnostic" not in fm.fields
        assert fm.fields["replicated_layout"] is True
        # 文件名以折叠后的 _r0 开头
        assert fm.get_file_name(make_offload_key(b"\x01" * 8, 0)).startswith(
            f"{fm.base_path}_r0/"
        )

```

tests/v1/kv_offload/tiering/test_fs_tier.py

端到端跨 TP 字节级 round-trip 测试，直接验证 FS tier 在 replicated_layout 下的正确性。

```

def test_fs_tier_cross_tp_round_trip(tmp_path):
    """TP=2 replicated writer 和 TP=4 reader 共享 namespace 和字节数据。"""

```

```

root = str(tmp_path)
writer_tensor = _page_aligned_rand_tensor(4, _BLOCK_ELEMENTS)
expected = writer_tensor[0].clone()
# 以 TP=2 写入
writer = FileSystemTierManager(
    offloading_spec=_make_offloading_spec(
        tp_size=2, world_size=2, rank=0, replicated_layout=True
    ),
    primary_kv_view=memoryview(writer_tensor.numpy()),
    tier_type="fs",
    root_dir=root,
    n_read_threads=2,
    n_write_threads=2,
)
try:
    writer.submit_store(make_job(1, [key(7)], [0]))
    assert all(r.success for r in drain(writer))
    writer_base = writer.file_mapper.base_path
    writer_path = writer.file_mapper.get_file_name(key(7))
finally:
    writer.shutdown()

reader_tensor = _page_aligned_zero_tensor(4, _BLOCK_ELEMENTS)
# 以 TP=4 读取, 应命中同一路径
reader = FileSystemTierManager(
    offloading_spec=_make_offloading_spec(
        tp_size=4, world_size=4, rank=3, replicated_layout=True
    ),
    primary_kv_view=memoryview(reader_tensor.numpy()),
    tier_type="fs",
    root_dir=root,
    n_read_threads=2,
    n_write_threads=2,
)
try:
    assert reader.file_mapper.base_path == writer_base
    assert reader.file_mapper.get_file_name(key(7)) == writer_path
    assert lookup_and_wait(reader, [key(7)]) == [LookupResult.HIT]
    reader.submit_load(make_job(2, [key(7)], [1], is_promotion=True))
    assert all(r.success for r in drain(reader))
    # 验证读取的数据与写入一致
    assert torch.allclose(reader_tensor[1], expected)
finally:
    reader.shutdown()

```

评论区精华

无实质性讨论。claude[bot] 自动评论（来自 fork 无法 review），orozery 直接批准。

- 暂无高价值评论线程

风险与影响

- 风险:

1. 兼容性风险（预期）：replicated_layout 会改变 base_path 的 hash（因为 fields 新增了 replicated_layout: True），导致已有 FS/OBJ 条目变冷，P2P 握手失败。PR 明确指出并提供 discriminator 隔离，属于预期行为。
2. 错误配置风险：如果 replicated_layout=True 但 parallel_agnostic=False，则不会折叠，与普通行无异。测试 test_replicated_layout_requires_caller_opt_in 覆盖此路径。
3. 与 parallel_agnostic 交互：replicated 行会与 parallel-agnostic 行走相同折叠路径，但通过 replicated_layout 字段区分，不会意外混用。测试 test_replicated_and_parallelism_agnostic_separate_layouts 验证。- 影响：对用户：启用 compact MLA (replicated_layout) 的场景（当前仅限 bare MLA 组、TP>1、单节点）可在不同 TP 大小间共享二级 KV 缓存，减少存储需求并加速多 TP 部署的重叠启动。对系统：二级缓存层（FS、OBJ、P2P）的 identity 计算统一在 FileMapper 中，各 tier 管理无需修改。P2P 兼容性由 get_run_config() 指纹保证，不同 replicated 配置会自然隔离。对团队：合并后需关注新 namespace 的产生，确保生产环境迁移时清理旧缓存或接受冷启动。

- 风险标记：兼容性变更，现有缓存失效，需调用者 opt-in

关联脉络

- PR #49152 [KV-offload][FS] : Batch store/load_block in C: 同为 KV offload 二级缓存层优化，该 PR 为批量 IO 下沉，本 PR 为跨 TP 身份折叠，共同完善 secondary-tier 能力。
- PR #49947 [Bugfix][KV Offload][OBJ] Preserve job completion during cleanup: 影响 OBJ 层作业完成与清理，本 PR 的 OBJ 测试也验证了跨 TP 身份折叠，两者配合确保正确性。