

PR #55879 完整报告

vllm-project/vllm

[Bugfix][Attention] Stabilize sparse-MLA DCP for GLM PCP evals

合并时间: 2026-09-16 00:58

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55879>

执行摘要

- 一句话: 修复稀疏 MLA 在 DCP/PCP 组合下的崩溃与卡死
- 推荐动作: 值得精读。该 PR 是稀疏 MLA 在 DCP/PCP 组合下稳定性修复的标本案例: 建议重点学习 `supports_combination` 的能力契约设计 (显式拒绝不支持的组合并给出可执行的回退建议)、Triton JIT warmup key 的覆盖策略 (归一化 key + 对齐笛卡尔积)、以及用 monkeypatch 把 `torch.tensor` 替换为抛错函数来证明热路径无设备标量构造的回归测试手法。注意 PR 依赖 open 的 #50791 且携带其原始作者提交 (thegoldenflow), 后续 main 上 #50791 的合并方式需要跟进, 避免重复提交或署名冲突。

功能与动机

PR body 明确目标为 "Make the B200 GLM-5.2 PCP evaluation complete reliably across its TP/PCP/DCP configurations"。主构建 #89033 先后暴露两个独立的 memory-budget 失败: TP2/PCP2 在 FlashInfer CUTLASS autotuning 需要 12.22 GiB 而仅剩约 7.6 GiB; TP1/PCP4/DCP4 在 CUDA graph profiling 后得到负的 3.23 GiB KV cache 预算。而关联 issue #50791 指出 #50781 的根因: FlashInfer 的 trtllm-gen launcher 在请求 LSE 时会从固定 workspace 中划出 softmax-stats slab, "a single request with a moderately long prompt is enough to bring down every DCP worker and the API server", 且该 workspace 对 DCP、head 数、batch 数完全无感知。

实现拆解

变更分六步推进, 每步由一次 exact Buildkite gate 暴露并验证:

1. 移植并适配 DCP 感知的 workspace 精确 sizing (`vllm/v1/attention/backends/mla/flashinfer_mla_sparse.py`): 携带 open PR #50791 中 thegoldenflow 的原始提交, 新增纯 Python 函数 `compute_trtllm_sparse_mla_workspace_bytes`, 按 trtllm-gen 内核公式 $\text{sizeof(float2)} * \text{allgather_heads} * \text{batch} * 256 + 1 \text{ MiB}$ 字节级复现软 max slab; `_required_workspace_bytes` 解析最终大小并尊重 `VLLM_FLASHINFER_WORKSPACE_BUFFER_SIZE` override (低于计算值时告警); `_get_workspace_buffer` 增加 `min_bytes` 参数; `FlashInferMLASparseTRTLLMMetadataBuilder.__init__` 在 DCP 下、首次 `forward/capture` 前完成预分配。适配点是只作用于 SM100 TRTLLM builder, SM120 独立后端保持原行为。

2. 调整评测内存预算与 scheduler cap (两个 tests/evals/gsm8k/configs/*.yaml) :
TP2/PCP2 设 --gpu-memory-utilization 0.85 给 CUTLASS autotuning 留约 12.5 GiB;
TP1/PCP4/DCP4 设 0.96 (高于 runtime 建议的 0.9545) 保证 KV 预算为正, 并把
--max-num-batched-tokens 从 32768 降到 4096, 使 64 heads/rank、DCP4 下的
workspace 恰好锁定为 2.39 GiB 而非 16.39 GiB。
3. 修复空 DCP 温启 rank (vllm/v1/attention/ops/dcp.py) : mask_dcp_empty_shards_ 在
seq_lens 为空时把 padded LSE 所有行填 -inf 并提前返回, 避免旧逻辑把序列索引钳到
-1 后索引空张量。
4. 扩展 indexer 的 Triton warmup key (vllm/v1/attention/backends/mla/indexer.py) :
BuildPrefillChunkMetadataKernel.get_warmup_keys 在 PCP+DCP 时用 zip_inputs 额外
预热归一化 key (DCP_RANK=0, DCP_WORLD=1); input_variant 由 2 种对齐组合扩展为
uncompressed_seq_lens 与 cu_compressed_seq_lens 各自对齐 / 不对齐的 4 种笛卡尔
积, 覆盖 PCP 打包分配导致的非 16 字节对齐指针。
5. 拒绝不支持的 FlashInfer 后端组合 (flashinfer_mla_sparse.py 的
supports_combination) : 当 prefill_context_parallel_size > 1 且
decode_context_parallel_size > 1 时返回明确错误, 自动选择回退到能先 gather DCP
KV shard 的 FLASHMLA_SPARSE; PCP-only 与 DCP-only 组合继续放行。
6. 消除 decode 热路径 GPU 同步 (indexer.py + vllm/v1/attention/backends/utils.py) :
metadata builder 初始化时物化一次 dcp_rank_tensor (int32 设备标量) ,
_dcp_localize_decode_seq_lens 改传该张量; get_dcp_local_seq_lens 的 dcp_rank 参数
扩展为 int | torch.Tensor | None, Tensor 分支断言 dtype、device、numel 后直接复用
, 避免 build() 中每次构造 host-to-device 标量触发 runtime 的 GPU<->CPU sync guard。

测试配套: 新增 tests/v1/attention/test_flashinfer_sparse_mla_workspace.py (166 行, 字节精确复现 #50781 报告值、env override 语义、后端组合允许 / 拒绝) ;

test_indexer_dcp_localize.py 用 monkeypatch 让 torch.tensor 抛错以证明热路径复用 rank 张量; test_indexer_deepseek_v4_slot_mapping.py 验证归一化 key 与 4 种对齐组合; test_dcp_a2a_pack_mask.py 覆盖空 rank 并把 CUDA 用例标记 requires_accelerator。

关键文件:

- vllm/v1/attention/backends/mla/flashinfer_mla_sparse.py (模块 MLA 后端; 类别 source; 类型 core-logic; 符号 compute_trtllm_sparse_mla_workspace_bytes, _required_workspace_bytes, _get_workspace_buffer, FlashInferMLASparseTRTLLMBackend.supports_combination) : 核心修复文件: DCP 感知的 workspace 字节级公式、TRTLLM metadata builder 预分配、PCP+DCP 后端组合拒绝。
- vllm/v1/attention/backends/mla/indexer.py (模块 稀疏索引器; 类别 source; 类型 core-logic; 符号 BuildPrefillChunkMetadataKernel.get_warmup_keys, BuildPrefillChunkMetadataKernel.warmup_inputs, FlashInferMLASparseIndexerMetadataBuilder.init, FlashInferMLASparseIndexerMetadataBuilder._dcp_localize_decode_seq_lens) : PCP+DCP 归一化 warmup key、4 种指针对齐组合、dcp_rank_tensor 物化, 解决首请求 JIT 不同步与 decode 热路径 GPU sync。

- `vllm/v1/attention/backends/utils.py` (模块 DCP 工具; 类别 source; 类型 core-logic; 符号 `get_dcp_local_seq_lens`) : `get_dcp_local_seq_lens` 扩展为接受设备张量 rank, 是消除 decode 热路径 GPU<->CPU 同步的关键。
- `vllm/v1/attention/ops/dcp.py` (模块 DCP 算子; 类别 infra; 类型 core-logic; 符号 `mask_dcp_empty_shards_`) : `mask_dcp_empty_shards_` 处理零本地序列的空 shard, 修复 CUDA graph warmup 下索引空张量的崩溃。
- `tests/v1/attention/test_flashinfer_sparse_mla_workspace.py` (模块 workspace 测试; 类别 test; 类型 test-coverage; 符号 `test_reporter_softmax_carve_is_reproduced_byte_exactly`, `test_reporter_config_covers_observed_overflow`, `test_non_dcp_size_is_unchanged`, `test_combined_pcp_dcp_rejects_flashinfer_sparse`) : 新增 166 行 CPU 回归测试: 字节级复现 #50781 报告值、env override 语义、后端组合允许 / 拒绝。
- `tests/v1/attention/test_indexer_deepseek_v4_slot_mapping.py` (模块 索引器测试; 类别 test; 类型 test-coverage; 符号 `test_indexer_warmup_includes_pcp_normalized_dcp_key`) : 验证 warmup key 集合同时包含配置的 DCP rank 与归一化 (0, 1), 且覆盖 4 种指针对齐组合。
- `tests/v1/attention/test_indexer_dcp_localize.py` (模块 DCP 定位测试; 类别 test; 类型 test-coverage; 符号 `test_get_dcp_local_seq_lens_reuses_device_rank_tensor`, `fail_tensor_construction`) : 用 monkeypatch 让 torch.tensor 抛错, 证明 decode 定位热路径直接复用初始化时物化的 rank 张量。
- `tests/v1/attention/test_dcp_a2a_pack_mask.py` (模块 mask 测试; 类别 test; 类型 test-coverage; 符号 `test_eager_mask_handles_rank_with_no_local_sequences`) : 新增空 rank 掩码回归, 并把 GPU 相关用例用 `requires_accelerator` 隔离。
- `tests/evals/gsm8k/configs/GLM-5.2-NVFP4-TP1-PCP4-DCP4-EP.yaml` (模块 评测配置; 类别 test; 类型 configuration) : 评测配置 0.96 利用率 + 4096 scheduler cap, 把 DCP workspace 约束到 2.39 GiB 并保证 KV 预算为正。
- `tests/evals/gsm8k/configs/GLM-5.2-NVFP4-TP2-PCP2-EP.yaml` (模块 评测配置; 类别 test; 类型 configuration) : 评测配置 0.85 利用率给 CUTLASS autotuning 预留约 12.5 GiB 余量。

关键符号: `compute_trtllm_sparse_mla_workspace_bytes`, `required_workspace_bytes`, `_get_workspace_buffer`, `FlashInferMLASparseTRTLLMBackend.supports_combination`, `FlashInferMLASparseTRTLLMMetadataBuilder.__init__`, `BuildPrefillChunkMetadataKernel.get_warmup_keys`, `BuildPrefillChunkMetadataKernel.warmup_inputs`, `get_dcp_local_seq_lens`, `mask_dcp_empty_shards`, `FlashInferMLASparseIndexerMetadataBuilder._dcp_localize_decode_seq_lens`

关键源码片段

`vllm/v1/attention/backends/mla/indexer.py`

PCP+DCP 归一化 warmup key、4 种指针对齐组合、`dcp_rank_tensor` 物化, 解决首请求 JIT 不同步与 decode 热路径 GPU sync。

```

def get_warmup_keys(self, vllm_config: VllmConfig) -> list[CompileKey]:
    max_tokens = max(1, min(vllm_config.scheduler_config.max_num_batched_tokens, 8))
    hf_text_config = vllm_config.model_config.hf_text_config
    parallel_config = vllm_config.parallel_config
    dcp_world = parallel_config.decode_context_parallel_size
    dcp_interleave = parallel_config.cp_kv_cache_interleave_size
    dcp_rank = get_dcp_group().rank_in_group if dcp_world > 1 else 0

    # PCP+DCP 的 prefill 会构造全局 PCP chunk plan, 本内核收到的行起点
    # 已是本地化结果, 运行时 key 被归一化为 rank 0/world 1;
    # 必须显式预热该 key, 否则某个 rank 会在首次真实 batch 时触发 JIT,
    # 而其它 rank 已进入 collective, 导致 all-gather 挂起超时。
    dcp_cases = [dict(DCP_RANK=dcp_rank, DCP_WORLD=dcp_world)]
    if parallel_config.prefill_context_parallel_size > 1 and dcp_world > 1:
        dcp_cases.append(dict(DCP_RANK=0, DCP_WORLD=1))

    compress_ratios = tuple(
        dict.fromkeys(
            max(1, int(ratio))
            for ratio in (
                *(getattr(hf_text_config, "compress_ratios", None) or (1,)),
                getattr(hf_text_config, "index_kpool", 1) or 1,
            )
        )
    )
    index_kpool = getattr(hf_text_config, "index_kpool", None)
    if index_kpool and index_kpool > 1 and index_kpool not in compress_ratios:
        compress_ratios = compress_ratios + (index_kpool,)

    return self._trace_dispatch(self.dispatch)(
        zip_inputs(*dcp_cases), # 同时覆盖配置的 rank 与归一化的 (0, 1) 两种 key 族
        # 覆盖 Triton 的 divisible、exact-one 与 generic i32 三类标量。
        query_slice_start=(0, 1, 2),
        query_slice_stop=(1, 2 * max_tokens - 1, 2 * max_tokens),
        DCP_INTERLEAVE=dcp_interleave,
        BLOCK_SIZE=self.BLOCK_SIZE,
        COMPRESS_RATIO=list(compress_ratios),
        # PCP 的全局累计长度是 packed 多行分配里的第二行, 指针不一定
        # 16 字节对齐; prefill chunking 也能独立地错位 uncompressed 长度,
        # 因此对两个指针分别预热对齐 / 不对齐共 4 种 input_variant。
        input_variant=(
            TritonPointerInputVariant.from_alignment(
                uncompressed_seq_lens=True, cu_compressed_seq_lens=True,
            ),
            TritonPointerInputVariant.from_alignment(
                uncompressed_seq_lens=True, cu_compressed_seq_lens=False,
            ),
            TritonPointerInputVariant.from_alignment(
                uncompressed_seq_lens=False, cu_compressed_seq_lens=True,

```

```

    ),
    TritonPointerInputVariant.from_alignment(
        uncompressed_seq_lens=False, cu_compressed_seq_lens=False,
    ),
),
)

```

vllm/v1/attention/backends/utils.py

get_dcp_local_seq_lens 扩展为接受设备张量 rank，是消除 decode 热路径 GPU<->CPU 同步的关键。

```

def get_dcp_local_seq_lens(
    seq_lens: torch.Tensor,
    dcp_size: int = 1,
    dcp_rank: int | torch.Tensor | None = None,
    cp_kv_cache_interleave_size: int = 1,
) -> torch.Tensor:
    """在 DCP 下把全局 decode seq_lens 切分为本 rank 的本地长度。

    各 DCP rank 上的 KV cache 大小可能不同，需要按 rank 偏移计算本地
    seq_lens。dcp_rank 既可以传普通 int（会构造设备标量，仅用于非热路径），
    也可以传已经物化的 int32 设备标量，直接复用避免 GPU<->CPU 同步。
    """
    seq_lens_i32 = seq_lens.to(torch.int32)

    if dcp_rank is None:
        # 不指定 rank 时返回全部 rank 的展开结果，供统一计算使用。
        rank_offsets = torch.arange(
            dcp_size, dtype=torch.int32, device=seq_lens.device,
        ).view(*(1,) * seq_lens_i32.dim()), dcp_size)
        seq_lens_tiled = seq_lens_i32.unsqueeze(-1)
    elif isinstance(dcp_rank, torch.Tensor):
        # 热路径：直接复用 builder 初始化时物化的标量，必须与 seq_lens
        # 同 dtype、同设备且为单元素，否则说明调用方传错对象。
        assert dcp_rank.dtype == torch.int32
        assert dcp_rank.device == seq_lens.device
        assert dcp_rank.numel() == 1
        rank_offsets = dcp_rank
        seq_lens_tiled = seq_lens_i32
    else:
        # 非热路径：int 会被包成设备标量（会引入一次 host-to-device 同步）。
        rank_offsets = torch.tensor(
            dcp_rank, dtype=torch.int32, device=seq_lens.device,
        )
        seq_lens_tiled = seq_lens_i32

    base = (
        seq_lens_tiled
        // cp_kv_cache_interleave_size

```

```

    // dcp_size
    * cp_kv_cache_interleave_size
)
remainder = seq_lens_tiled - base * dcp_size
remainder = torch.clip(
    remainder - rank_offsets * cp_kv_cache_interleave_size,
    0,
    cp_kv_cache_interleave_size,
)
return base + remainder

```

评论区精华

该 PR 的 review 正文为空 (fork 的自动 review 被禁用, tlrnchlsmith 直接 APPROVED), 核心讨论都发生在 issue 评论与 Buildkite gate 里:

tlrmchlsmith: "merged main to pick up <https://github.com/vllm-project/vllm/pull/55499>, which I think was breaking the CI for this PR"

khluu 的多轮 gate 归因: 明确把 #89050 归为 main 分支运行而非 PR 分支证据, "I am not treating it as PR-branch gate evidence"; 每次新增 commit 后提醒旧 approval 视为 stale, 直到 final exact gate #89097 三配置全部通过。

NCCL 定位的根因交锋: "rank 0 entering the LSE all-gather with count 9,792 while ranks 1-3 entered with 10,240—the exact 153 vs 160 local query rows x 64 heads mismatch", 由此确认 FLASHINFER_MLA_SPARSE 不做跨 DCP KV gather, 设计结论是让 supports_combination 拒绝组合并回退 FLASHMLA_SPARSE。

decode 热路径的 sync guard: "get_dcp_local_seq_lens called torch.tensor(dcp_rank, device=seq_lens.device) from metadata build(), and the runtime guard rejected that per-batch host-to-device scalar construction", 最终采用初始化时物化 + monkeypatch 回归测试的方案。

- exact gate 归因与 stale approval 管理 (testing): 以最终 exact gate #89097 为准, 三组配置全部通过; 验证流程严格区分 literal-head、actual-branch 与诊断运行。
- 零本地序列 DCP warmup rank 的掩码行为 (correctness): mask_dcp_empty_shards_ 在 seq_lens 为空时直接把所有 padded LSE 行填 -inf 并提前返回。
- PCP+DCP 下 FlashInfer sparse MLA 的不等 all-gather 挂起 (design): supports_combination 拒绝 PCP+DCP 组合并给出可执行错误, 自动选择回退到实现所需 gather 的 FLASHMLA_SPARSE; CPU 回归覆盖组合拒绝与单模式放行。
- decode 热路径的 GPU<->CPU 同步 guard (performance): 在 builder 初始化时物化 dcp_rank_tensor, 热路径直接复用; 回归测试用 monkeypatch 让 torch.tensor 抛错来证明无替换构造。
- 合并 main 引入 #55499 修复 CI (other): 已合并 main, 后续 gate 正常执行。

风险与影响

- 风险:

1. workspace 预分配与显存占用: `compute_trtllm_sparse_mla_workspace_bytes` 按 `max_num_batched_tokens` 计算, 普通用户在 DCP 下若 `scheduler cap` 较大 (如 32768 tokens), workspace 会膨胀到 16.39 GiB 量级, 可能挤占 KV cache 或直接 OOM; 本次用评测配置降 `cap` 规避, 但生产部署需要意识到该行为变化。
2. CUDA graph 地址烘焙: workspace 大小在首次 capture 前必须一次到位 (代码注释明确 "no lazy regrow"), 若未来有路径在 capture 后扩大 batch 上限, 会触发溢出。
3. 后端自动回退: PCP+DCP 下 FLASHINFER_MLA_SPARSE 被拒绝后自动选 FLASHMLA_SPARSE; 若运行环境未安装 FlashMLA 依赖, 可能导致启动失败。
4. warmup 集合扩大: indexer 的编译 key 从 2 变 4 种对齐组合并新增一组 DCP key, 首次启动编译时间与显存占用略增; 对极端 kernel 数量可能拉长启动。
5. 验证覆盖: 新增回归均为 CPU 单测, GPU 行为只由一次性 Buildkite 手工 gate 验证 (最终 #89097 通过), 常规 CI 对 DCP/PCP 组合无持续覆盖。
6. 评测配置贴近显存上限: TP1/PCP4/DCP4 用 0.96 利用率, 若 B200 空闲显存有波动仍可能 OOM。
 - 影响: 用户影响: 所有在 B200/SM100 上使用 FLASHINFER_MLA_SPARSE + DCP (或 PCP+DCP) 的用户。此前单条约 3K token 以上的 prompt 就能让整个 DCP worker 组崩溃 (#50781), 本 PR 从 workspace 层面根治; 同时消除了首个真实 batch 因 JIT 编译不同步导致的 all-gather 挂起, 以及 decode 步的 GPU 同步报错。系统影响: 启动时 DCP 下会增加 workspace 预分配 (与 batch 上限挂钩); PCP+DCP 组合的后端选择从 FlashInfer 稀疏 MLA 自动切换到 FlashMLA 稀疏, 注意力计算路径发生变化。团队影响: GLM-5.2 的 TP/PCP/DCP 评测从不可复现的失败变为可复现的通过 (TP2/PCP2 0.9121、TP1/PCP4 0.9454、TP1/PCP4/DCP4 0.8832, 全部 1319 题、invalid 0), 并为后续并行注意力调试沉淀了一套 "字节级公式 + exact gate 逐层归因 + monkeypatch 防回归" 的方法论。
 - 风险标记: 核心路径变更, workspace 预分配增大显存占用, PCP+DCP 后端自动回退行为变化, 依赖 open PR #50791 的移植, GPU 验证仅靠一次性 Buildkite gate, 评测配置接近显存上限

关联脉络

- PR #50791 [Bugfix][Attention] Size FlashInfer sparse MLA workspace for decode-context-parallel: 本 PR 的直接来源: workspace 字节级 sizing 原始诊断与实现来自 #50791 (修复 #50781), 本 PR 携带其原始作者提交 (thegoldenflow) 并适配当前 main 的分离 SM120 后端。
- PR #56677 [CI] Select the supported DCP backend for PCP eval: PR body 明确依赖其 ag_rs 后端修复: exact main build #89033 确认该修复已生效, 才暴露出后续两类 memory-budget 失败。
- PR #56825 [Bugfix][SparseMLA] Fix piecewise cudagraph capture crash in index group: 同属 v1 稀疏 MLA + indexer + CUDA graph 区域的修复, 反映该功能线在 cudagraph 与并行组合下的持续加固。
- PR #56969 [Bugfix] Prevent out-of-bounds access in FlashInfer SM90 sparse MLA mixed batches: 同为 FlashInfer sparse MLA 后端的健壮性修复 (SM90 侧), 与本文

SM100/DCP 侧修复构成同一后端矩阵的稳定性演进。