PR #29775 完整报告

sgl-project/sglang

[DeepSeek V4] Enable FlashMLA sparse prefill by default

合并时间：2026-07-02 04:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29775

执行摘要

- 一句话：默认启用 FlashMLA sparse prefill，GB300 预填充吞吐提升 4-12%
- 推荐动作：建议所有 DeepSeek V4 部署升级此 PR，以自动获得预填充吞吐提升。在关键生产环境升级前，建议在目标 GPU（尤其非 GB300 架构）上运行性能回归测试。合并者和作者已充分测试 GB300 场景，但其他环境需自行验证。

功能与动机

FlashMLA sparse prefill 能显著提升 DeepSeek V4 的预填充吞吐，且已在之前版本中实现但默认关闭。PR body 中测试数据显示，在饱和负载下，1K/8K/16K/128K 输入长度分别获得 +4.2%、+6.8%、+8.0%、+12.4% 的吞吐提升。为让所有用户自动受益，将此特性默认启用。

实现拆解

实现拆解为以下 5 步：

1. 修改环境变量默认值：在 python/sglang/srt/environ.py 中将 SGLANG_OPT_FLASHMLA_SPARSE_PREFILL 从 EnvBool(False) 改为 EnvBool(True)，实现默认启用。
2. 引入workspace管理器：新增SparsePrefillWorkspace类（位于sparse_prefill_utils.py），为稀疏预填充提供可增长、可复用的 scratch 存储，避免每层重新分配。该类按需调整 buffer 大小，只增不减。
3. 改进缓存空间计算：SparsePrefillChunkCache 新增 max_seq_len 字段，ensure_c4 和 ensure_c128 方法基于实际序列长度（而非 CUDA graph 捕获时的最大 padding）分配 token ID 数组，减少内存浪费。同时移除了原先分散的 c0_workspace、c4_workspace、c128_workspace，统一由 SparsePrefillWorkspace 管理。
4. 后端集成：在 deepseek_v4_backend.py 的 DeepseekV4AttnBackend 中创建 self.sparse_prefill_workspace，并在 _forward_prefill_sparse 中使用它替代原有的 workspace 引用。同时修改 DSV4Metadata.copy_ 和 refresh_for_breakable_cuda_graph_replay_ 以正确重置缓存，确保 CUDA graph 回放时重新构建。
5. Context Parallelism 兼容处理：在 deepseek_v4_hook.py 的 validate_deepseek_v4_cp 中，当启用 CP 时自动设置 SGLANG_OPT_FLASHMLA_SPARSE_PREFILL=False 并记录警告，因为 sparse prefill 当前未与 CP 兼容。

6. 测试覆盖：在 test_deepseek_v4.py 中新增三个测试：test_sparse_prefill_workspace_reuses_and_grows 验证 workspace 复用与增长；test_sparse_prefill_c4_uses_live_extent 和 test_sparse_prefill_c128_uses_live_extent 验证基于实际序列长度的空间计算。同时加固了 TestDSV4BreakableCudaGraphMetadataContract 中关于 sparse_prefill_cache 重置的断言。

关键文件：

- python/sglang/srt/layers/attention/dsv4/sparse_prefill_utils.py（模块 预填充工具；类别 source；类型 core-logic；符号 SparsePrefillWorkspace, init, get）：引入 SparsePrefillWorkspace 类，统一管理 workspace；修改 SparsePrefillChunkCache 使用 max_seq_len 以基于实际序列长度计算空间，移除了分散的 workspace 字段。
- test/registered/attention/unittests/dsv4/test_deepseek_v4.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_sparse_prefill_cache, test_sparse_prefill_workspace_reuses_and_grows, test_sparse_prefill_c4_uses_live_extent, test_sparse_prefill_c128_uses_live_extent）：新增三个单元测试覆盖 workspace 复用与增长、C4/C128 使用实际序列长度；加固了 CUDA graph metadata 重置测试。
- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 注意力后端；类别 source；类型 core-logic）：集成 SparsePrefillWorkspace，在 Backend 中创建并传入；修改 _forward_prefill_sparse 使用统一 workspace；重置 cache 逻辑适配 CUDA graph。
- python/sglang/srt/arg_groups/deepseek_v4_hook.py（模块 参数配置；类别 source；类型 dependency-wiring）：在 context parallelism 启用时自动禁用 FlashMLA sparse prefill，避免不兼容。
- python/sglang/srt/environ.py（模块 环境变量；类别 source；类型 core-logic）：修改默认值，是功能默认启用的入口点。

关键符号：SparsePrefillWorkspace.init, SparsePrefillWorkspace.get, SparsePrefillChunkCache.init, DeepseekV4AttnBackend._forward_prefill_sparse

关键源码片段

python/sglang/srt/layers/attention/dsv4/sparse_prefill_utils.py

引入 SparsePrefillWorkspace 类，统一管理 workspace；修改 SparsePrefillChunkCache 使用 max_seq_len 以基于实际序列长度计算空间，移除了分散的 workspace 字段。

```
class SparsePrefillWorkspace:
    """Backend-owned scratch storage for sparse prefill KV dequantization.

    The workspace contents are fully overwritten before every attention call,
    so token buckets and compression ratios can safely share one buffer. Sparse
    prefill executes eagerly and serially on the supported paths, which makes it
    safe to replace the scratch allocation when a larger extent is needed.
    """

    def __init__(self, device: torch.device):
```

```
        self.device = device
        self._buffer: Optional[torch.Tensor] = None

    def get(self, num_tokens: int) -> torch.Tensor:
        assert num_tokens > 0
        # 当前 buffer 容量，若为 None 则视为 0
        current_capacity = self._buffer.shape[0] if self._buffer is not None else 0
        # 只在需要更大空间时才重新分配，否则复用
        if num_tokens > current_capacity:
            self._buffer = torch.empty(
                (num_tokens, 1, WORKSPACE_DIM),
                dtype=torch.bfloat16,
                device=self.device,
            )
        # 返回所需长度的视图（允许复用更大 buffer）
        return self._buffer[:num_tokens]
```

评论区精华

本 PR 无实质 review 讨论环节，合入者 Fridge003 直接批准。评论内容主要是 CI 重跑命令和结果确认，无设计争议。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 性能退化风险：虽然在 GB300 上测试有显著提升，但未在其他 GPU（如 SM120）或非饱和负载场景验证，可能存在性能回退。PR 作者已明确指出这部分超出当前安全验证范围。
  2. 默认行为变更：现有用户若依赖旧行为（sparse prefill 关闭）可能遇到意外变化，但可通过设置 SGLANG_OPT_FLASHMLA_SPARSE_PREFILL=0 恢复。
  3. Context Parallelism 兼容：已通过自动禁用处理，但若用户同时设置环境变量强制启用 CP 和 sparse prefill，可能导致异常。需要确保 CP 路径正确执行。
  4. CUDA graph 交互：共享 workspace 在 CUDA graph 捕获 / 重放场景下的正确性依赖于 sparse_prefill_cache 的重置逻辑，测试已覆盖但仍有风险。 - 影响：用户影响：所有 DeepSeek V4 用户自动获得预填充性能提升（4-12%），无需修改配置。显式关闭的用户行为不受影响。Context parallelism 用户不受影响（自动禁用）。系统影响：略微增加运行时内存（workspace buffer），但总体更高效。环境变量依赖代码 envs.SGLANG_OPT_FLASHMLA_SPARSE_PREFILL 的使用者需注意默认值变化。团队影响：为后续 sparse prefill 相关优化（如 CP 兼容、SM120 适配）提供了更清晰的代码基础。
- 风险标记：默认行为变更，context parallelism 自动禁用，未在 SM120 测试

关联脉络

- 暂无明显关联 PR