

PR #38293 完整报告

sgl-project/sglang

[CP V1 Deprecation 3.5/5] Deprecate HIP/NPU/MUSA prefill CP and remove legacy implementation

合并时间: 2026-09-08 06:39

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38293>

执行摘要

- 一句话: 废弃 HIP/NPU/MUSA 平台 prefill CP, 移除旧实现, 为 CP 重构铺路。
- 推荐动作: 该 PR 是一个重要且必要的架构清理步骤, 值得精读以理解 CP 模块的演进方向。其核心价值在于揭示了从平台特定实现向统一策略模式迁移的路径。关键设计决策包括: 1) 通过早期参数验证实现优雅降级而非运行时失败; 2) 保留导入存根以最小化对保留后端的影响, 同时标记为“错误提示”; 3) 配套全面的单元测试以保证行为契约。

功能与动机

为即将进行的 CP 架构重构做准备。PR body 明确指出: “Deprecate the legacy prefill context-parallel implementations on HIP/ROCm, Ascend NPU, and MUSA ahead of the upcoming CP refactor. CUDA's strategy-based prefill CP, decode context parallelism (DCP), and ordinary non-CP platform inference remain in scope for regression protection, not removal.” 其动机是清理历史遗留代码, 统一 CP 实现, 消除平台特定的维护负担。

实现拆解

1. 入口守卫: 在 `arg_groups/parallel_hook.py` 中新增 `validate_prefill_cp_platform` 函数, 在服务器启动早期 (模型下载和虚拟模型短路之前) 拒绝 HIP/NPU/MUSA 上的 `--enable-prefill-cp` 请求, 并抛出明确的弃用错误。同时移除了 `handle_platform_cp_compatibility` 和 `handle_legacy_cp_runtime_compatibility` 中为这些平台提供的旧参数别名和兼容性投影。
2. 移除旧运行时字段与逻辑: 从 `model_executor/forward_batch_info.py` 等文件中移除了支持旧平台 CP 的四个运行时字段。在 `layers/utils/cp_utils.py` 中, 删除了 `ContextParallelMetadata` 数据类、`prepare_context_parallel_metadata` 等核心函数, 以及所有相关的 all-gather 重排、分片和重建逻辑, 仅保留错误提示的导入存根。
3. 清理模型侧 CP 代码: 在多个模型文件 (`deepseek_v4.py`, `deepseek_v2.py`, `deepseek_nextn.py`, `deepseek_v4_nextn.py`, `qwen2_moe.py`, `qwen3_moe.py`) 中, 移除了针对 HIP/NPU/MUSA 平台的 CP 元数据准备、token 分片 / 重建、KV 缓存重建等代码路径。这些模型现在只依赖通用的 CP 策略 (通过 `get_cp_strategy()`)。
4. 移除重复通信器与重叠逻辑: 从 `distributed/parallel_state.py` 中移除了 `attn_cp_overlap` 通信器及其初始化 / 销毁逻辑, 因为这是为平台特定 CP 重叠准备的。在 `dsa_backend.py`

和 `layers/attention/dsv4/compressor.py` 中移除了相关的异步 all-gather 启动和完成逻辑。

5. 测试配套：新增 `test/registered/unit/server_args/test_platform_prefill_cp_deprecation.py`，包含针对平台守卫逻辑的全面单元测试，覆盖各种平台、策略组合，确保弃用行为正确且不影响非 CP 或 DCP 路径。更新了 `test_server_args.py` 中的现有测试以反映新的行为。

关键文件：

- `python/sglang/srt/models/deepseek_v4.py`（模块 DeepSeek V4 模型；类别 source；类型 core-logic；符号 `_cp_tbo_launch`, `op_cp_gather_a`, `op_cp_gather_b`, `op_cp_moe`）：核心模型文件，移除了约 300 行与平台 CP+TBO 重叠相关的逻辑，包括操作调度、元数据管理和通信启动 / 完成函数。
- `python/sglang/srt/layers/utils/cp_utils.py`（模块 CP 工具层；类别 source；类型 dependency-wiring；符号 `ContextParallelMetadata`, `is_prefill_context_parallel_enabled`, `is_mla_prefill_cp_enabled`, `mla_use_prefill_cp`）：CP 工具函数的核心文件，从包含 678 行的完整实现缩减为仅 23 行的导入存根，删除了 `ContextParallelMetadata`、所有 all-gather 重排、分片 / 重建函数。
- `python/sglang/srt/arg_groups/parallel_hook.py`（模块 参数验证钩子；类别 source；类型 core-logic；符号 `handle_platform_cp_compatibility`, `handle_legacy_cp_runtime_compatibility`, `validate_prefill_cp_platform`）：服务器参数解析和验证的关键入口点，新增了平台弃用守卫，并移除了为平台 CP 提供的兼容性逻辑。
- `test/registered/unit/server_args/test_platform_prefill_cp_deprecation.py`（模块 弃用验证测试；类别 test；类型 test-coverage；符号 `TestPlatformPrefillCPDeprecation`, `test_platform_cp_rejected_before_model_lookup`, `test_context_parallel_handler_rejects_before_model_lookup`, `test_resolution_rejects_even_dummy_models`）：为新的平台弃用逻辑提供了全面的单元测试，覆盖多种平台和参数组合。
- `python/sglang/srt/layers/attention/dsa_backend.py`（模块 DSA 注意力后端；类别 source；类型 core-logic；符号 `_all_gather_dsa_trtllm_fp8_kv`, `materialize_full_kv_cp`）：DSA 注意力后端的核心文件，移除了平台特定的 CP KV 材化、all-gather 和分片逻辑，统一使用通用策略。

关键符号：`validate_prefill_cp_platform`, `handle_context_parallelism`, `is_dsa_enable_prefill_cp`, `is_dsa_prefill_cp_round_robin_split`, `_can_run_tbo`, `materialize_full_kv_cp`, `_all_gather_dsa_trtllm_fp8_kv`

关键源码片段

`python/sglang/srt/models/deepseek_v4.py`

核心模型文件，移除了约 300 行与平台 CP+TBO 重叠相关的逻辑，包括操作调度、元数据管理和通信启动 / 完成函数。

```
# 以下代码片段展示了 _can_run_tbo 方法在移除平台 CP 路径后的简化逻辑。
# 变更前，该方法包含对 HIP 平台 CP round-robin-split 路径的特殊处理。
# 变更后，它只检查非 CP 路径和通用 TBO 使能条件。
def _can_run_tbo(self, forward_batch: ForwardBatch) -> bool:
    """DSV4 prefill-only two-batch-overlap gate."""
```

```

from sglang.srt.layers.moe import is_tbo_enabled
# 移除了对 dsa_use_prefill_cp(forward_batch) 的复杂检查，因为平台 CP 已废弃。
# 现在的逻辑更清晰：只允许非 CP 的预填充路径运行 TBO。
if not forward_batch.forward_mode.is_extend_without_speculative():
    return False
# ... ( 省略 PP 检查等其他条件 ) ...
return is_tbo_enabled() and self._non_cp_tbo_eligible(forward_batch)

```

python/sglang/srt/arg_groups/parallel_hook.py

服务器参数解析和验证的关键入口点，新增了平台弃用守卫，并移除了为平台 CP 提供的兼容性逻辑。

```

def validate_prefill_cp_platform(server_args: Any):
    """Reject deprecated platform CP before resolving models or CP topology."""
    cfg = resolving_view(server_args)
    platform = get_platform()
    # 在模型加载和拓扑设置之前，拒绝目标平台上的 prefill CP 请求。
    if cfg.enable_prefill_cp and (
        platform.is_hip or platform.is_npu or platform.is_musa
    ):
        raise ValueError(
            "Prefill CP on HIP/NPU/MUSA is deprecated; CP support will be refactored soon."
        )

# 该函数现在作为 handle_context_parallelism 的第一行调用。
def handle_context_parallelism(server_args: Any):
    validate_prefill_cp_platform(server_args)
    # ... ( 后续的常规验证逻辑 ) ...

```

评论区精华

无正式的 review 讨论线程。PR 仅包含 2 条 issue 评论，均为机器人重跑测试的命令和结果。这表明变更主要是机械性的清理，可能在合并前已在其他渠道（如 Slack 或设计文档）达成共识。

- 暂无高价值评论线程

风险与影响

- 风险：
 1. 平台功能回归：对 HIP/NPU/MUSA 平台使用 `--enable-prefill-cp` 的用户将立即遇到启动错误。这是预期行为，但需要在发布说明中明确通知。
 2. 残留依赖：为保留的独立平台后端文件（如 `hardware_backend/npu/attention/ascend_backend.py`）保留了错误提示的导入存根。如果这些存根的函数签名在未来被意外修改，可能导致导入错误。然而，这是为后续清理（如 PR 描述中提到的“delete together with E”）而有意为之的过渡状态。

3. 测试覆盖：新测试主要覆盖 CPU 上的参数验证逻辑。完整的运行时 CI 测试（包括 GPU/NPU 上的精度和端到端测试）尚未运行，存在未验证的集成风险。PR 作者已声明此 PR 为草稿，等待 CI 和平台审查。- 影响：用户影响：直接影响所有在 HIP/NPU/MUSA 平台上尝试使用 `--enable-prefill-cp` 的用户，他们将收到明确的弃用错误。CUDA 用户 and 所有平台的 DCP/ 非 CP 用户不受影响。系统影响：显著减少了代码库的复杂性和维护负担，消除了数百行平台特定的重复逻辑和分支，使 CP 代码路径更加清晰，为后续统一重构奠定了基础。团队影响：简化了未来的 CP 功能开发，因为不再需要维护多套平台特定的实现。

- 风险标记：平台功能回归，测试覆盖不足（GPU/NPU 运行时），残留依赖存根

关联脉络

- PR #33480 [AMD] Support prefill context parallel two batch overlap for DeepSeek V4: 本 PR 的一个核心目标是审计并移除 PR #33480 中添加的、为平台特定 CP 服务的 TBO 重叠代码（如重复通信器、操作调度、元数据重建）。
- PR #38389 [Scheduler] Unify per-iteration request intake into `ingest_requests()`: 同属近期调度器重构系列，虽然功能不同，但都致力于简化调度器逻辑和统一代码路径，与本 PR 的清理和重构目标在精神上的一致。
- PR #37143 [Scheduler] Make request-timeout aborts rank-consistent to fix TP collective hangs: 另一个涉及调度器和并行逻辑的修复 PR，强调了在多 rank 环境中保持逻辑一致性的重要性，这与本 PR 移除可能导致不一致的平台特定分支的目标相关。