PR #33614 完整报告

sgl-project/sglang

[Spec] Fix Dspark and Dflash state divergence across TP rank

合并时间：2026-08-30 08:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33614

执行摘要

- 一句话：广播 rank 0 采样决策，修复 DSpark/DFlash TP 分歧死锁
- 推荐动作：值得精读。这是投机解码 TP 一致性的系统性修复，展示了三类有价值的设计决策：CUDA graph 内执行 collectives 的约束与回退策略；' 在决策派生任何状态前广播 ' 比 ' 在消费点同步 ' 更安全的设计原则；用环境变量按站点裁剪做线上故障 bisection 的方法论。特别值得学习评论区中 hnyls2002 的两次纠错——一次要求补 budget 同步、一次基于纯函数推导撤销该要求并实测 50 us host-blocking 开销，构成性能关键路径上 ' 最小同步集 ' 的推导与验证闭环。阅读时可重点对比 commit 1c1e458（回到普通 broadcast）与 050df8e（移除 budget 同步）背后的权衡。

功能与动机

issue #33289 描述了 2× DGX Spark（GB10）上 DeepSeek-V4 + DSpark 的间歇性死锁：rank A 卡在 NCCL proxy 的 logits all-gather 中，rank B 空转在 _broadcast_reqs_across_ranks，watchdog 最终杀掉进程。PR body 给出的机理是：为维持投机加速，SGLang 默认跳过采样 token 的跨 rank 同步；DSpark 每个 rank 独立做三类决策（draft Markov 链逐 token 采样、目标 verify 推导 correct_len/bonus/cap_trim_lens、prefill 采样 next_token_ids），一旦某个 rank 提交了不同的 token 或接受长度，序列长度与 KV 状态开始漂移，一个无法匹配的 collective 会永久卡死 NCCL proxy。作者明确解释不选择对齐 RNG seed 的原因：rank 可能消费不同数量的随机值，且无法覆盖非 RNG 差异。社区多位用户（Han-xin58、icewool、tampajohn）确认 0.5.16/0.5.17 仍存在该问题，tampajohn 在 Qwen3.8-27B NVFP4 + DFlash2 上以 8 并发采样流约 50% 复现率复现，证实不是 DeepSeek 专属。

实现拆解

1. 新增统一同步基础设施 python/sglang/srt/speculative/spec_tp_sync.py：SpecTpSyncSite 枚举定义 16 个广播站点（DSpark 11 个 + DFlash 5 个），覆盖草案采样、规划、验收、目标输出、显存门控；SpecTpSync 类封装 broadcast（TP=1 时 no-op，但始终解析环境变量，让任何部署上的配置拼写错误在启动期暴露）；parse_spec_tp_sync 支持 all/rng/init/off 预置与站点 slug/ 数字混用（- 前缀取反），为故障二分提供开关。
2. DSpark 全链路注入（dspark_worker_v2.py 为组装点）：构造期在 DP attention 下选择 attn_tp_group、否则 tp_group 创建 SpecTpSync，注入 planner/proposer/verify

epilogue/draft sampler；dspark_draft.py 的 sample_draft_block 在 greedy/sample/multinomial 三条草案采样路径逐 step 广播（DSPARK_DRAFT_）；dspark_draft_sampler.py 在 graph 折叠采样的 sampler 闭包内广播（DSPARK_GRAPH_SAMPLE/GREEDY），因为每步依赖前一个 token，只在 block 末尾同步太晚；dspark_planner.py 把 verify_lens 广播迁移到 SpecTpSync（DSPARK_PLAN）并删除旧 verify_lens_broadcast_group；dspark_verify.py 在 eager 与 captured epilogue 的 finalize、token 输出、KV 提交前同步 correct_len/bonus/cap_trim_lens 三元组（DSPARK_ACCEPT_）；prefill 输出 next_token_ids 在进入草案路径前广播（DSPARK_TARGET）；AUTO folded-sampling 与图捕获的显存探测改用组最小可用显存（DSPARK_MEM），保证各 rank 折叠 / 捕获决策一致。

3. DFlash 扩展（commit fb8b202）：dflash_worker_v2.py 提取 _accept_block，在 selector 采样、非 greedy 采样验证、greedy argmax 三条路径分别同步 accept_len/bonus/target_predict（DFLASH_SELECTOR/ACCEPT_SAMPLE/ACCEPT_GREEDY），prefill next_token_ids 在 DFLASH_TARGET 广播，draft graph 捕获前显存探测改用组最小（DFLASH_MEM）。
4. 配置与诊断（environ.py）：注册 SGLANG_SPEC_TP_SYNC（默认 all），可按站点裁剪同步用于线上隔离与 bisection（issue 评论中已完成 195 分钟无挂起的二分验证）。
5. 测试与验证配套：PR 最终文件列表未包含独立测试文件；验证主要依赖 fault-injection 复现（注入 rank 1 的 accept_len 偏移）、GSM8K 精度对比、DFlash A/B 性能对比、外部用户 cherry-pick 验证，以及合并前 /rerun-test 触发的 dspark/dflash sanity 测试（全部通过）。讨论中作者提到曾补充 13/13 回归测试（分歧 budget 注入），但未进入最终文件列表。

关键文件：

- python/sglang/srt/speculative/spec_tp_sync.py（模块 投机解码；类别 source；类型 core-logic；符号 SpecTpSyncSite, SpecTpSync, parse_spec_tp_sync, _resolve）：新增的统一 TP 同步基础设施：SpecTpSyncSite 枚举定义全部 16 个决策点，SpecTpSync 是广播封装，SGLANG_SPEC_TP_SYNC 解析器支持按站点裁剪做故障二分，是本次修复的骨架。
- python/sglang/srt/speculative/dflash_worker_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 _accept_block, forward_batch_generation, init_cuda_graphs, _maybe_build_draft_sampler）：DFlash 的验收 / 接受逻辑重构为 _accept_block，三条 accept 路径（selector 采样、非 greedy 采样验证、greedy argmax）分别注入 DFLASH_SELECTOR/ACCEPT_SAMPLE/ACCEPT_GREEDY 广播，prefill next_token_ids 在 DFLASH_TARGET 广播，graph capture 前显存探测改为组最小。
- python/sglang/srt/speculative/dspark_components/dspark_draft_sampler.py（模块 投机解码；类别 source；类型 core-logic；符号 sampler, _resolve_folded_sampling, DsparkDraftSampler.call, maybe_build_draft_sampler）：graph 折叠采样路径的核心：在 sampler 闭包内逐 step 广播草案 token（DSPARK_GRAPH_SAMPLE/GREEDY），并把 folded-sampling AUTO 决策的显存探测改为组最小，保证捕获时所有 rank 折叠一致。
- python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 DSparkWorkerV2.init,

_maybe_build_draft_sampler, init_cuda_graphs, _forward_prefill）：DSpark worker 的组装点：构造 SpecTpSync（DP attention 下用 attn_tp_group），注入 planner/proposer/verify epilogue/draft sampler，prefill next_token_ids 在进入草案路径前广播（DSPARK_TARGET），并将 decode graph 判断与显存探测改为组最小。

- python/sglang/srt/speculative/dspark_components/dspark_planner.py（模块 投机解码；类别 source；类型 core-logic；符号 _schedule_verify_lens, resolve_verify_token_budget, verify_lens_broadcast_group）：verify_lens 规划决策从旧的 verify_lens_broadcast_group 迁移到 SpecTpSync（DSPARK_PLAN），并移除 resolve_verify_token_budget 中多余的 per-step CPU budget 广播（约 50 us host-blocking 且为纯函数派生，已在 review 中自我纠正后删除）。
- python/sglang/srt/speculative/dspark_components/dspark_verify.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 TargetVerifyExecutor.accept_and_finalize, DsparkVerifyEpilogue._accept）：eager 与 captured epilogue 两条验收路径在 finalize、token 输出、KV 提交前同步 correct_len/bonus/cap_trim_lens 三元组（DSPARK_ACCEPT_GREEDY/SAMPLE/GRAPH），是 KV 状态分叉的最后防线。
- python/sglang/srt/speculative/dspark_components/dspark_draft.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 sample_draft_block, DraftBlockProposer.init, DraftBlockProposer.propose）：草案采样入口：greedy/sample/multinomial 三条逐 step 采样路径在返回前广播草案 token，保证后续 verify 依赖一致。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration；符号 EnvStr）：新增 SGLANG_SPEC_TP_SYNC 环境变量（默认 all），支持按站点裁剪同步用于线上隔离与 bisection，是诊断与规避手段的入口。

关键符号：SpecTpSync.sync, SpecTpSync.enabled, SpecTpSync.available_memory_gb, parse_spec_tp_sync, dflash_worker_v2._accept_block, DsparkDraftSampler.call, _resolve_folded_sampling, DSparkWorkerV2._maybe_build_draft_sampler, DSparkVerifyPlanner._schedule_verify_lens, TargetVerifyExecutor.accept_and_finalize, DsparkVerifyEpilogue._accept, sample_draft_block, DraftBlockProposer.propose

关键源码片段

python/sglang/srt/speculative/spec_tp_sync.py

新增的统一 TP 同步基础设施：SpecTpSyncSite 枚举定义全部 16 个决策点，SpecTpSync 是广播封装，SGLANG_SPEC_TP_SYNC解析器支持按站点裁剪做故障二分，是本次修复的骨架。

```
# python/sglang/srt/speculative/spec_tp_sync.py（新增文件）

class SpecTpSyncSite(IntEnum):
    """每个投机决策广播点：DSpark 11 个 + DFlash 5 个站点。

    站点编号与 slug 是 SGLANG_SPEC_TP_SYNC 的稳定选择句柄，
    支持按站点开启/关闭与 "-" 取反，用于故障二分定位真实分歧点。
    """
```

```
    # -- DSpark --
    DSPARK_MEM = 1 # 显存门控：同时决定 graph capture 与 folded-sampling
    DSPARK_DRAFT_GREEDY = 2
    DSPARK_DRAFT_SAMPLE = 3
    DSPARK_DRAFT_MULTINOMIAL = 4
    DSPARK_GRAPH_SAMPLE = 5 # 图内 philox 噪声采样，每次 replay 重画
    DSPARK_GRAPH_GREEDY = 6
    DSPARK_PLAN = 7 # verify_lens 规划结果
    DSPARK_ACCEPT_GREEDY = 8
    DSPARK_ACCEPT_SAMPLE = 9
    DSPARK_ACCEPT_GRAPH = 10 # captured verify epilogue 内的 accept 结果
    DSPARK_TARGET = 11 # prefill next_token_ids

    # -- DFlash --
    DFLASH_MEM = 12
    DFLASH_SELECTOR = 13
    DFLASH_ACCEPT_SAMPLE = 14
    DFLASH_ACCEPT_GREEDY = 15
    DFLASH_TARGET = 16

    @property
    def slug(self) -> str:
        return self.name.lower().replace("_", "-")


# 预置集合：all / off / init（仅初始化显存类站点）/ rng（init + 所有采样类站点）。
_PRESETS = {
    "all": _ALL,
    "off": frozenset(),
    "none": frozenset(),
    "init": _INIT,
    "rng": _INIT | _RNG,
}

# 支持预置名、slug、数字；"-" 前缀取反；逗号分隔，便于线上逐步缩小同步范围。
def parse_spec_tp_sync(spec: str) -> frozenset[SpecTpSyncSite]:
    sites: frozenset[SpecTpSyncSite] = frozenset()
    for token in spec.replace(" ", "").replace("_", "-").lower().split(","):
        if not token:
            continue
        negate = token.startswith("-")
        value = _resolve(token[1:] if negate else token)
        sites = sites - value if negate else sites | value
    return sites


class SpecTpSync:
    """把 rank 0 的投机决策广播给 TP 组；TP=1 时为空操作。
```

```
    即使单卡也解析环境变量，让任何部署上的拼写错误在启动期暴露。
    """

    def __init__(self, tp_group) -> None:
        self._tp_group = tp_group
        sites = parse_spec_tp_sync(envs.SGLANG_SPEC_TP_SYNC.get())
        self._sites = sites if tp_group.world_size > 1 else frozenset()
        if sites != _ALL and tp_group.world_size > 1 and tp_group.rank_in_group == 0:
            logger.warning(
                "Speculative TP sync limited to %s.",
                [f"{int(s)}:{s.slug}" for s in sorted(sites)] or "no site",
            )

    def enabled(self, site: SpecTpSyncSite) -> bool:
        return site in self._sites

    def sync(self, site: SpecTpSyncSite, values: torch.Tensor) -> torch.Tensor:
        # 只有站点启用且 TP > 1 时才真正广播，其余情况是零拷贝直通。
        if site in self._sites:
            self._tp_group.broadcast(values, src=0)
        return values

    def available_memory_gb(self, site, device, gpu_id, *, group):
        """返回组内最小可用显存：保证所有 rank 的 capture/folded 决策一致。

        如果各自探测本地显存，AUTO 模式可能在部分 rank 折叠、部分不折叠，
        同一批请求就会走上不同的执行路径。
        """
        distributed = self.enabled(site) and group.world_size > 1
        return get_available_gpu_memory(
            device,
            gpu_id,
            distributed=distributed,
            cpu_group=group.cpu_group if distributed else None,
        )
```

python/sglang/srt/speculative/dspark_components/dspark_draft_sampler.py

graph 折叠采样路径的核心：在 sampler 闭包内逐 step 广播草案 token（DSPARK_GRAPH_SAMPLE/GREEDY），并把 folded-sampling AUTO 决策的显存探测改为组最小，保证捕获时所有 rank 折叠一致。

```
# 关键片段：CUDA graph 内逐 step 采样的跨 rank 同步（dspark_draft_sampler.py）。
# 草案 Markov 链每一步都依赖前一个 token，只在 draft block 末尾同步太晚；
# 必须把广播折叠进每一次采样的输出，保证所有 rank 的下一步输入一致。
if self.folded_sampling:

    def sampler(step_logits: torch.Tensor, step_idx: int) -> torch.Tensor:
        del step_idx
        # 图内 philox 噪声：每次 graph replay 推进生成器并重画噪声，
```

```
        # 不同 rank 即使 logits 相同也可能采出不同 token，必须广播。
        noise = self.exp_noise[:bs].exponential_()
        return self._tp_sync.sync(
            SpecTpSyncSite.DSPARK_GRAPH_SAMPLE,
            SampleStepTokens.execute(
                step_logits=step_logits,
                temperatures=self.temperatures[:bs],
                greedy_mask=self.greedy_mask[:bs],
                exp_noise=noise,
            ),
        )

    else:
        # greedy 折叠路径同样需要同步：rank 间浮点噪声或分片边界
        # 同样可能让 argmax 结果不同。
        def sampler(step_logits: torch.Tensor, step_idx: int) -> torch.Tensor:
            return self._tp_sync.sync(
                SpecTpSyncSite.DSPARK_GRAPH_GREEDY,
                greedy_step_sampler(step_logits, step_idx),
            )
```

评论区精华

hnyls2002 对初版方向的评价与 A/B 要求：'Direction is right, and the fault-injection demo is a good way to pin the failure mode... gamma+3 broadcasts per decode step sit on the critical path and can't overlap with compute; that needs a number next to it.' 作者承认 'pre-fix baseline 无法存活 ' 的表述是在发现 NCCL 问题之前写的、更新描述时漏改，随后补做 DFlash 四对 AB/BA A/B 与 GSM8K 精度对比。

hnyls2002 关于复用 DSA graph-safe 广播：'DSA indexer already solves this: _broadcast_indexer_topk_from_rank0... under --enable-dp-attention with attn_tp_size < tp_size, attn_tp_group is built with use_pynccl=... and both are off by default. Can this reuse that shape instead of adding a second one?' 作者先实现 capture-safe 广播与 attn-tp PyNCCL provisioning，hnyls2002 随后引用 #36963 指出现状已过时、PyNCCL margin 很小（256B 下 7.17 vs 7.64 us），最终整体回退为普通 broadcast 并删除 provisioning（commit cbe65e7 / 1c1e458）。

hnyls2002 对 ' 消费点同步无安全网 ' 的质疑与自我纠正是最精彩的一轮：先质疑 verify_token_budget 未同步；作者补了同步；hnyls2002 最终承认 'I asked for this sync in an earlier round and I was wrong -- the budget is already a pure function of values that agree across ranks... It cost a one-element gloo broadcast on the CPU group every decode step -- ~50 us on 2x H200, host-blocking, and it only ran under the overlap scheduler it was defeating. Removed in 050df8e.' 这是性能关键路径上 ' 最小同步集 ' 推导与验证的闭环。

simple-sun 指出 DFlash 同样问题并建议同 PR 修复，作者注入 accept_len skew 复现后提交 fb8b202；simple-sun 还指出 SGLANG_SIMULATE_ACC_LEN 模拟接受路径（默认禁用）仍可独立分歧，属已知边界。外部验证方面，tampajohn 报告 'Cherry-picked #33614 onto

our image... 9/9 clean load-test runs across DSpark/DFlash2 × bf16/fp8, zero hangs,throughput unchanged'。

- DsparkTpSync 应复用 DSA graph-safe 广播而非另起炉灶 (design): 作者先实现 broadcast_capture_safe 与 attn-tp PyNCCL provisioning；hnyls2002 进一步用 #36963 的回退现状简化，最终删除 capture-safe helper 与 provisioning，统一走普通 broadcast（commit 1c1e458 / cbe65e7）。
- 消费点同步无安全网：verify_token_budget 是否需要广播 (design): budget 同步在 commit 050df8e 中移除，验证 ' 最小同步集 ' 应基于纯函数推导而非消费点枚举。
- 性能声明需要 A/B 与关键路径广播开销量化 (performance): DFlash 输出吞吐差异在 ±0.26 tok/s 内、TPOT 差异 ±0.25 ms 内，精度 0.960 与 target-only 持平；DSpark 无 pre-fix 基线（修复前无法在该负载下存活）。
- SGLANG_SIMULATE_ACC_LEN 模拟接受路径的剩余分歧点 (correctness): 该路径默认禁用；作者确认关注但最终未纳入同步，属已知边界。
- SGLANG_SPEC_TP_SYNC 站点裁剪的 bisection 验证 (testing): 确认 token 同步是修复核心，也验证了诊断开关的有效性；Dflash 二分仍在进行。
- DFlash 同步扩展与外部用户验证 (testing): 扩展被接受并合入，跨模型与 fp8 KV 场景验证通过。

风险与影响

- 风险：
  1. 性能风险：decode 关键路径新增 gamma+3 次 [bs] 张量广播且无法与计算重叠；DFlash A/B 在 TP=2、24 条 prompt 下显示差异在噪声内（±0.26 tok/s），但更大 TP 规模、更小 batch、更高并发下的影响未被测量。
  2. 覆盖缺口：PR 未附带测试文件，核心路径（decode step 循环）依赖手工 fault-injection 与外部用户验证，缺少自动化回归保护。
  3. 图捕获兼容性：captured graph 内执行 broadcast 依赖 TP communicator 的 PyNCCL 可用性；最终方案删除了 capture-safe helper 与 attn-tp PyNCCL provisioning，若某环境 TP 组无 PyNCCL 且图内走广播，会在 capture 时暴露。
  4. 已知边界：SGLANG_SIMULATE_ACC_LEN 模拟接受路径（默认禁用）每 TP 进程可独立采样强制提交长度，仍可导致 accept_len/commit_lens/seq len 分歧；用户若误设 SGLANG_SPEC_TP_SYNC=off 会恢复死锁。
  5. 部署依赖：修复在 NCCL 2.30.7 下验证（需 SGLANG_NCCL_SO_PATH 注入），torch 自带 2.28.9 在该工作负载下仍会 wedge，修复有效性部分依赖外部 NCCL 版本。- 影响：对用户：修复 issue #33289 描述的间歇性多节点 TP 死锁，覆盖 DSpark 与 DFlash 两族投机解码算法、eager 与 CUDA graph 折叠两条执行路径；DFlash 吞吐无回退，DSpark 修复后 31.9 tok/s 稳定运行（TP=2，DeepSeek-V4-Flash-0731），多个外部用户复现与验证通过。对系统：新增 16 个同步站点与按站点裁剪开关，成为投机解码 TP 一致性的可复用基础设施；' 决策在派生任何状态前广播 ' 成为新的跨 rank 不变量。对团队：23 个 commit 的长周期 PR，初版独立 DsparkTpSync 与 PyNCCL provisioning 在 review 中被大幅简化（普通 broadcast），体现 ' 最小同步集 ' 设计导

向。 - 风险标记：核心路径新增广播（decode step），无自动化测试文件，图捕获依赖 TP PyNCCL，外部 NCCL 版本依赖，模拟接受路径已知分歧边界

关联脉络

- PR #31041 [Spec] Add LFM2 and LFM2-MoE DSpark speculative decoding support: 同属 DSpark 投机解码功能线（models/dspark、fused_kv_write 等），本 PR 为该功能线补齐多节点 TP 一致性基础；PR body 亦引用同族 EAGLE TP 同步 issue #29003 与 PR #31478 的思路。
- PR #36897 Decouple speculative draft capacity from runtime state: 同属 speculative 子系统基础设施演进，重构 runtime_context/spec_info 的运行时状态；与本 PR 的 SpecTpSync 初始化上下文（get_tp_group/get_parallel）处于同一模块，反映投机解码模块持续治理方向。