PR #34133 完整报告

sgl-project/sglang

config: derive the runner's DCP topology from its ParallelState

合并时间：2026-08-09 16:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34133

执行摘要

- 一句话：DCP 拓扑改由 ParallelState 派生，统一为 attn_dcp_* 命名
- 推荐动作：值得精读。变更虽只有 +15/-15，但包含三个有价值的设计决策：(1) 进程级事实与 per-worker 状态的边界划分——get_parallel() 无法表达 draft 差异，而显式传入的 ParallelState 可以；(2) 构造时序约束下的 rank 推导（tp_rank % dcp_size）及其与既有 attn_cp_rank 等字段的一致性规约，commit 历史中“引入再删除 compute_dcp_rank”的演进尤其值得注意；(3) 与 #32858 的职责互补关系（KV-head 布局 vs runner 拓扑）。建议结合 #32858 与 #33925 一起阅读，能完整理解 SGLang DCP 并行状态建模的演进脉络。

功能与动机

前序 PR #33925 把 ModelRunner.dcp_size / .dcp_rank 移到 init_torch_distributed() 之后读取活拓扑，解决了“何时读”的问题，但数值仍是进程级事实贴在 per-worker 对象上：每个进程只有一个 DCP 组，get_parallel().attn_dcp_size 在结构上无法区分 target 与 draft，而 draft 绝不能切分 token 维度——这正是 #32858 描述的失败模式（draft KV 池按 target 的 dcp_size=4 超分 4 倍）。PR body 明确论证：'There is one DCP group per process, so get_parallel().attn_dcp_size is structurally incapable of differing between the target and a draft'，而 runner 已有的 ps（ParallelState）是逐 worker 显式传入的冻结结构，draft 在其中已声明 pp_rank=0，具备表达差异的能力。

实现拆解

实现按以下 4 步展开：

1. 数据契约调整（parallel_state_wrapper.py）：ParallelState 中的裸 dcp_size 字段重命名为 attn_dcp_size，移入 attn_* 块并紧邻 attn_cp_size，同时新增 attn_dcp_rank 字段；trivial() 默认值同步改为 1 / 0。理由是与 attn_tp、attn_cp、attn_dp 等其他注意力并行轴保持 rank/size 成对命名规约，并且和 get_parallel().attn_dcp_* 这一无断言读取路径的命名对齐。
2. 构造点内联推导（scheduler.py、benchmark/one_batch.py）：两个 ParallelState 构造点都传入 attn_dcp_size=server_args.dcp_size、attn_dcp_rank=tp_rank % server_args.dcp_size。推导依据是 DCP 组按 TP 组内连续 dcp_size 切片构建（bootstrap.py的组构造现在消费ps.attn_dcp_size），组内序号恒为tp_rank%dcp_size。由于 ParallelState 在 Scheduler.__init__ 中构建、早于 init_torch_distributed，它本身是组构造的输入，因此 rank 不可能从 live group 读回，只能内联推导——这与

attn_cp_rank、moe_dp_rank、moe_ep_rank 的既有做法一致（commit 历史中曾短暂引入 compute_dcp_rank 辅助函数，随后因破坏"与其他 rank 一致"的规约而被回退为内联表达式）。

3. 消费端切换：model_runner.py 删除 self.dcp_size / self.dcp_rank 两行赋值（类内已无引用）；tp_worker.py 的 alloc_memory_pool 与 get_worker_info 中 max_req_len 改为乘 self.ps.attn_dcp_size，确保 KV 池与 worker 信息使用本 worker 的并行状态；eager_runner.py 的 _execute_extend 与 forward_batch_info.py 的 DCP KV mask 计算改读 model_runner.ps.attn_dcp_size / attn_dcp_rank，后者顺带删除了 getattr(model_runner, "dcp_size", 1) 防御读取——因为 test/kits/attention_unittest 的 mock runner 均已设置 self.ps = ParallelState.trivial()，无需 fallback。
4. 测试配套：未新增单元测试文件，依赖既有 DCP *e2e* 回归（test/registered/dcp/*）。PR 内通过 /rerun-test 在 8-gpu-h200 与 4-gpu-b200 上重跑 DCP 测试；bootstrap.py 仅跟随字段重命名（ps.dcp_size → ps.attn_dcp_size）。

关键文件：

- python/sglang/srt/distributed/parallel_state_wrapper.py（模块 并行状态；类别 source；类型 data-contract；符号 ParallelState, ParallelState.trivial）：数据契约核心：ParallelState 的 dcp_size 重命名为 attn_dcp_size 并移入 attn_* 块，新增 attn_dcp_rank 字段，trivial() 同步更新。这是本 PR 所有消费端改动的源头。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 Scheduler.init）：ParallelState 主构造点：新增 attn_dcp_rank=tp_rank % server_args.dcp_size 内联推导，是本 PR 将 DCP 拓扑写入 per-worker 状态的入口。
- python/sglang/srt/model_executor/model_runner.py（模块 模型运行；类别 source；类型 data-contract；符号 ModelRunner.init）：删除 self.dcp_size / self.dcp_rank 两行对进程级 get_parallel() 结果的拷贝，消除 process-global 事实在 per-worker 对象上的残留。
- python/sglang/srt/managers/tp_worker.py（模块 工作器；类别 source；类型 core-logic；符号 alloc_memory_pool, get_worker_info）：alloc_memory_pool 与 get_worker_info 中 max_req_len 的计算从 model_runner.dcp_size 改为本 worker 的 self.ps.attn_dcp_size，直接影响 KV 池容量上限，是本 PR 对 draft worker 差异化的关键落点。
- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向批次；类别 source；类型 data-contract；符号 init_new）：HIP DCP KV mask 的生成改读 model_runner.ps.attn_dcp_size / attn_dcp_rank，并移除 getattr(model_runner, "dcp_size", 1) 防御读取，依赖 mock runner 已设置 ps=ParallelState.trivial()。
- python/sglang/srt/model_executor/runner/eager_runner.py（模块 即时执行；类别 source；类型 data-contract；符号 _execute_extend）：DCP extend metadata 的准备条件从 model_runner.dcp_size 改为 model_runner.ps.attn_dcp_size，保证 eager 路径使用 per-worker 拓扑。
- python/sglang/srt/distributed/bootstrap.py（模块 分布式启动；类别 source；类型 core-logic；符号 init_torch_distributed）：DCP 组构造跟随字段重命名（ps.dcp_size → ps.attn_dcp_size），保证 parallel_state 与 torus 组构造的契约一致。

- python/sglang/benchmark/one_batch.py（模块 基准脚本；类别 source；类型 core-logic ；符号 load_model）：benchmark 场景的第二个 ParallelState 构造点，同步内联推导 attn_dcp_rank / attn_dcp_size，避免与主路径契约漂移。

关键符号：ParallelState, ParallelState.trivial, Scheduler.init, ModelRunner.init, alloc_memory_pool, get_worker_info, init_new, _execute_extend, init_torch_distributed

关键源码片段

python/sglang/srt/distributed/parallel_state_wrapper.py

数据契约核心：ParallelState 的 dcp_size 重命名为 attn_dcp_size 并移入 attn_* 块，新增 attn_dcp_rank 字段，trivial() 同步更新。这是本 PR 所有消费端改动的源头。

```
from dataclasses import dataclass
from typing import Optional


@dataclass(frozen=True, slots=True, kw_only=True)
class ParallelState:
    """每个 worker 显式传入的冻结并行拓扑记录。

    DCP 相关字段按 attn_* 命名规约放入前向注意力的并行块：
    attn_dcp_size 表示 DCP 组大小，attn_dcp_rank 表示本 worker 在其
    DCP 组内的序号。二者在 Scheduler.__init__ 中、init_torch_distributed
    之前直接由 server_args 与 tp_rank 推导——因为 DCP 组本身就是按
    TP 组内连续 dcp_size 切片构建的，所以组内序号恒为 tp_rank % dcp_size，
    无法（也无需）从 live group 读回。
    """

    tp_rank: int
    tp_size: int
    pp_rank: int
    pp_size: int
    dp_rank: Optional[int]
    dp_size: int
    attn_tp_rank: int
    attn_tp_size: int
    attn_cp_rank: int
    attn_cp_size: int
    attn_dcp_rank: int # DCP 组内序号，推导自 tp_rank % dcp_size
    attn_dcp_size: int # DCP 组大小，与 attn_* 系列命名对齐
    attn_dp_rank: int
    attn_dp_size: int
    moe_ep_rank: int
    moe_ep_size: int
    moe_dp_rank: Optional[int]
    moe_dp_size: int
    gpu_id: int
```

```
    @staticmethod
    def trivial(**overrides: Optional[int]) -> "ParallelState":
        """单卡 / 测试用默认值，DCP 一律取 1 / 0。"""
        kwargs: dict[str, Optional[int]] = dict(
            tp_rank=0,
            tp_size=1,
            pp_rank=0,
            pp_size=1,
            dp_rank=0,
            dp_size=1,
            attn_tp_rank=0,
            attn_tp_size=1,
            attn_cp_rank=0,
            attn_cp_size=1,
            attn_dcp_rank=0,
            attn_dcp_size=1,
            attn_dp_rank=0,
            attn_dp_size=1,
            moe_ep_rank=0,
            moe_ep_size=1,
            moe_dp_rank=0,
            moe_dp_size=1,
            gpu_id=0,
        )
        kwargs.update(overrides)
        return ParallelState(**kwargs)
```

python/sglang/srt/managers/scheduler.py

ParallelState 主构造点：新增 attn_dcp_rank=tp_rank % server_args.dcp_size 内联推导，是本 PR 将 DCP 拓扑写入 per-worker 状态的入口。

```
# Distributed rank info
attn_tp_rank, attn_tp_size, attn_dp_rank, attn_dp_size = (
    compute_dp_attention_world_info(
        server_args.enable_dp_attention,
        tp_rank,
        server_args.tp_size,
        server_args.dp_size,
        server_args.attn_cp_size,
    )
)
# ParallelState 是每个 worker 的显式冻结拓扑记录，也是 DCP 组构造的输入。
# 它必须在 init_torch_distributed 之前构建，因此 attn_dcp_rank 无法从
# live group 读取，只能按"DCP 组 = TP 组内连续 dcp_size 切片"的约定
# 用 tp_rank % dcp_size 推导——与 attn_cp_rank / moe_dp_rank 等
# 其他并行轴 rank 的内联推导方式保持一致。
self.ps = ParallelState(
    tp_rank=tp_rank,
    tp_size=server_args.tp_size,
```

```
        pp_rank=pp_rank,
        pp_size=server_args.pp_size,
        dp_rank=dp_rank,
        dp_size=server_args.dp_size,
        attn_tp_rank=attn_tp_rank,
        attn_tp_size=attn_tp_size,
        attn_cp_rank=attn_cp_rank,
        attn_cp_size=server_args.attn_cp_size,
        attn_dcp_rank=tp_rank % server_args.dcp_size,
        attn_dcp_size=server_args.dcp_size,
        attn_dp_rank=attn_dp_rank,
        attn_dp_size=attn_dp_size,
        moe_ep_rank=moe_ep_rank,
        moe_ep_size=server_args.ep_size,
        moe_dp_rank=moe_dp_rank,
        moe_dp_size=server_args.moe_dp_size,
        gpu_id=gpu_id,
    )
```

评论区精华

该 PR 没有独立 review 评论，核心讨论集中在 PR body 的设计论证与 issue 评论中的 CI 重跑过程：

- 设计论证（作者 kpham-sgl）：作者在 body 中详细论证了为什么不能在构造时从 live group 读取 rank——ParallelState 在 Scheduler.__init__ 中构建，早于 init_torch_distributed，且它本身就是组构造的输入。同时明确区分本 PR 与 #32858 的职责边界：'#32858 answers the KV-head half of it via is_draft_model'，'this PR is the runner's DCP topology'，两者不共享代码、可组合。
- draft 语义的遗留问题（未解决）：作者明确指出 draft 是否应声明自己的 attn_dcp_size=1（draft 是 TP-sharded、不切分 token 维度）是独立问题，且 triton_backend 仍在 init 时读取 get_parallel().attn_dcp_size，'Both are follow-ups, not this PR'。
- CI 重跑过程（testing）：作者两次提交 /rerun-test test/registered/dcp/*。第一次 4-gpu-b200 的 4 个测试整体失败；第二次整体重跑仍显示失败；随后单独重跑 test_kimi_linear_dcp4.py 与 test_kimi_linear_dcp_dspark4.py 通过。test_qwen3p5_triton_dcp.py 与 test_tokenspeed_mla_dcp_metadata.py 未出现在最终成功重跑中，状态未最终确认。
  - DCP e2e 测试重跑与 4-gpu-b200 的部分失败 (testing): 主要 DCP 回归（layout 单测、8-gpu-h200 两组、kimi_linear 两组）最终通过；但 test_qwen3p5_triton_dcp 与 test_tokenspeed_mla_dcp_metadata 未包含在最终成功重跑中，且 PR extra CI 状态为 ⚪，失败是否与本 PR 相关未最终确认。
  - draft 模型是否应声明独立的 attn_dcp_size=1 (design): 明确留作后续工作，不属于本 PR：'Both are follow-ups, not this PR'。本 PR 只负责把拓扑值放到正确的位置。

风险与影响

- 风险：
  1. 行为等价性依赖构造时序：target 侧正确性依赖 ps.attn_dcp_size == server_args.dcp_size 且与 get_parallel().attn_dcp_size 一致（body 论证：DCP 组仅当 dcp_size > 1 时安装，initialize_model_parallel 在非 HIP/CUDA 平台拒绝 dcp_size > 1）。若未来 _init_parallel_groups 的安装条件变化，两处可能失配。
  2. 双读取源并存：triton_backend 仍在 init 时读取 get_parallel().attn_dcp_size，与本 PR 的 ps.attn_dcp_size 形成两套事实。当前依赖 draft 的 ps 携带 target 值才保持一致；一旦后续让 draft 声明 attn_dcp_size=1，若不同步改 triton_backend 会出现内部不一致。
  3. 测试缺口：无新增单元测试，依赖既有 DCP e2e；且 4-gpu-b200 上 test_qwen3p5_triton_dcp.py、test_tokenspeed_mla_dcp_metadata.py 在最终重跑中未覆盖，PR extra CI 状态为 ⁉，可能与机器波动相关但未确认。
  4. KV 池大小计算：tp_worker.py 的 max_req_len 是 KV 池容量上限，若任何 worker 的 ps.attn_dcp_size 与实际组不一致，会导致池容量错算（正是 #32858 的 4 倍超分问题形态）。 - 影响：影响范围为使用 DCP 的部署路径：forward_batch_info.py 的 HIP DCP KV mask、eager_runner.py 的 DCP extend metadata 准备、tp_worker.py 的 KV 池 max_req_len 计算，以及 speculative decoding 中 draft worker 的并行状态读取。对 target 与 draft 当前行为均等价（作者在 body 中逐一论证），属纯重构，不改变精度与性能。长期看，它把"并行拓扑的单一来源"从进程全局迁移到 per-worker ParallelState，使 draft 与 target 差异化成为可能，与 #32858 的 KV-head 布局修复组合后可彻底消除 draft 池超分问题；同时为后续所有并行维度的状态建模确立了"冻结记录 + 构造期推导"的模式。 - 风险标记：无新增单元测试，4-gpu-b200 部分测试状态未最终确认，triton_backend 仍读进程全局值，双读取源并存

关联脉络

- PR #32858 Fix DCP KV head mapping for GQA models: 同一 DCP 正确性问题的另一侧：该 PR 修复 DCP 组内 KV head 映射（get_num_kv_heads 认识 dcp_size、is_draft_model 特殊处理保持 draft TP-sharded），本 PR 负责 runner 的 DCP 拓扑；PR body 明确说明两者不共享代码、可组合，合起来 draft 在 head 布局与 attn_dcp_size 上均为 TP-sharded。
- PR #33925 ModelRunner DCP 字段读取时机调整（标题未在材料中提供）：PR body 指明本 PR 是 #33925 的 follow-up：#33925 把 ModelRunner.dcp_size / .dcp_rank 移到 init_torch_distributed() 之后读取活拓扑，解决了读取时机问题；本 PR 进一步把值迁移到 per-worker ParallelState，解决进程级事实无法区分 target/draft 的结构性问题。