PR #31161 完整报告

sgl-project/sglang

Introduce ModelRunner.ps ParallelState

合并时间：2026-07-14 16:01

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31161

执行摘要

- 一句话：引入 ParallelState 统一并行度参数
- 推荐动作：值得精读。此 PR 展示了跨组件状态统一的设计模式：用数据类封装参数，减少接口表面积，便于后续内部重构。Review 中发现的 pp_size 问题应作为合并前修复项，或至少记录在文档中。建议团队在合并后运行 speculative 相关 CI 以确保正确性。

功能与动机

PR body 明确指出要“Mirror Scheduler.ps”，即在 ModelRunner 中创建一个与 Scheduler 一致的 ParallelState 对象，替代分散的 tp_rank、moe_ep_rank、pp_rank 等参数，以减少构造函数参数数量并降低传递错误风险，同时和数据并行团队保持一致的顶层设计。

实现拆解

1. 数据结构定义：在 parallel_state_wrapper.py 中为 ParallelState 数据类添加 dcp_size 字段，并新增 trivial 静态工厂方法，用于生成默认单卡状态的实例。
2. 更新 ModelRunner 构造函数：在 model_runner.py 中，将 __init__ 的 7 个独立参数（tp_rank, tp_size, moe_ep_rank, moe_ep_size, pp_rank, pp_size, dp_rank, attn_cp_rank, moe_dp_rank）替换为一个 ps: ParallelState 参数，并从 ps 读取各字段赋值给 self.*。同时新增 self.ps 保存对象引用。
3. 简化 init_torch_distributed：init_torch_distributed 接口从九个独立参数改为接受 ps 对象，内部解包到局部变量，函数体不变。
4. 调整 TpModelWorker 和 speculative worker：在 tp_worker.py 及所有 speculative worker（dflash_worker_v2.py, dspark_worker_v2.py, eagle_worker_v2.py, multi_layer_eagle_worker_v2.py, standalone_worker_v2.py）的构造函数中，将多个秩参数合并为单个 ps 参数；在创建 draft worker 时使用 replace(ps, pp_rank=0) 调整。
5. 更新基准测试：在 one_batch.py 中，调用 compute_dp_attention_world_info 后构造完整的 ParallelState 对象。
6. 配套测试调整：修改测试文件中的构造调用（如 test_weight_checker.py）以适配新签名。

关键文件：

- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 data-contract；符号 ModelRunner.init, ModelRunner.init_torch_distributed）：核心入口，构造函数签名从 7 个独立参数改为 ps 对象，是本次重构的核心。

- python/sglang/srt/distributed/parallel_state_wrapper.py（模块 分布式状态；类别 source；类型 core-logic；符号 ParallelState.dcp_size, ParallelState.trivial）：定义了 ParallelState 数据类及 trivial 工厂方法，是整个重构的核心数据契约。
- python/sglang/srt/distributed/bootstrap.py（模块 分布式启动；类别 source；类型 dependency-wiring；符号 init_torch_distributed）：init_torch_distributed 接口从九个独立参数改为接受 ps 对象，是分布式初始化的关键入口。
- python/sglang/srt/managers/tp_worker.py（模块 工作节点；类别 source；类型 dependency-wiring；符号 TpModelWorker.init）：TpModelWorker 构造函数同样从多个独立参数简化为 ps 对象，是工作节点创建 ModelRunner 的桥梁。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 dependency-wiring；符号 EagleWorkerV2.init）：EAGLE speculative worker 的构造函数参数同步精简，并在创建 draft worker 时使用 replace(ps, pp_rank=0)。
- python/sglang/benchmark/one_batch.py（模块 基准测试；类别 source；类型 dependency-wiring）：基准测试需要手动构造 ParallelState 对象，展示了完整字段的使用方式。

关键符号：ParallelState.trivial, ModelRunner.init, ModelRunner.init_torch_distributed, TpModelWorker.init, DFlashWorkerV2.init, DSparkWorkerV2.init, EagleWorkerV2.init, MultiLayerEagleWorkerV2.init, StandaloneWorkerV2.init

关键源码片段

python/sglang/srt/model_executor/model_runner.py

核心入口，构造函数签名从 7 个独立参数改为 ps 对象，是本次重构的核心。

```
# python/sglang/srt/model_executor/model_runner.py (head 版本 )
# 引入 ParallelState 后，构造函数从原来接收多个秩参数变为只接收一个 ps 对象

from sglang.srt.distributed.parallel_state_wrapper import ParallelState

class ModelRunner(ModelRunnerKVCacheMixin):
    def __init__(
        self,
        model_config: ModelConfig,
        mem_fraction_static: float,
        gpu_id: int,
        ps: ParallelState, # 统一并行状态，替代 tp_rank、moe_ep_rank、pp_rank 等
        nccl_port: int,
        server_args: ServerArgs,
        is_draft_worker: bool = False,
        req_to_token_pool: Optional[ReqToTokenPool] = None,
        token_to_kv_pool_allocator: Optional[BaseTokenToKVPoolAllocator] = None,
        memory_pool_config: Optional[MemoryPoolConfig] = None,
        draft_model_idx: Optional[int] = None,
    ):
        # ... 其他初始化 ...
```

```
        self.tp_rank = ps.tp_rank
        self.tp_size = ps.tp_size
        self.dcp_size = server_args.dcp_size # 注意：review 建议改为 ps.dcp_size
        self.dcp_rank = ps.tp_rank % self.dcp_size
        self.ps = ps # 保存 ps 用于后续
        self.moe_ep_rank = ps.moe_ep_rank
        self.moe_ep_size = ps.moe_ep_size
        self.dp_rank = ps.dp_rank
        self.attn_dp_size = ps.attn_dp_size
        self.pp_rank = ps.pp_rank
        self.pp_size = ps.pp_size
        self.attn_cp_rank = ps.attn_cp_rank
        self.attn_cp_size = ps.attn_cp_size
        self.moe_dp_rank = ps.moe_dp_rank
        self.moe_dp_size = ps.moe_dp_size
        # ...
```

python/sglang/srt/distributed/parallel_state_wrapper.py

定义了 ParallelState 数据类及 trivial 工厂方法，是整个重构的核心数据契约。

```
# python/sglang/srt/distributed/parallel_state_wrapper.py (head 版本 )
from dataclasses import dataclass
from typing import Optional


@dataclass(frozen=True, slots=True, kw_only=True)
class ParallelState:
    """统一存放所有并行度信息的不可变数据类，冻结且使用 __slots__ 以节省内存。"""

    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_dp_rank: int
    attn_dp_size: int
    moe_ep_rank: int
    moe_ep_size: int
    moe_dp_rank: Optional[int]
    moe_dp_size: int
    dcp_size: int # device communication parallelism (decomposed TP) size
    gpu_id: int

    @staticmethod
```

```
    def trivial(**overrides: Optional[int]) -> "ParallelState":
        """生成一个默认单卡状态的 ParallelState 实例，所有秩为 0 、大小为 1。

        overrides 可用于覆盖特定字段（如 dp_rank=None）。
        """
        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_dp_rank=0,
            attn_dp_size=1,
            moe_ep_rank=0,
            moe_ep_size=1,
            moe_dp_rank=0,
            moe_dp_size=1,
            dcp_size=1,
            gpu_id=0,
        )
        kwargs.update(overrides)
        return ParallelState(**kwargs)
```

评论区精华

Gemini-code-assist 自动 review 指出两个关键问题：

- Critical：在所有 speculative draft worker 中，复制 target worker 的 ParallelState 时只设置 pp_rank=0，但未同时设置 pp_size=1，可能导致 draft worker 错误尝试初始化流水线并行组而引发挂起或失败。建议改为 replace(ps, pp_rank=0, pp_size=1)。
- Medium：在 ModelRunner.__init__ 中，self.dcp_size = server_args.dcp_size 应改为 self.dcp_size = ps.dcp_size，以与其他所有从 ps 读取的大小字段保持一致。 两个建议均未被采纳，风险仍然存在。
- Speculative worker 缺少 pp_size=1 设置 (correctness): PR 未按建议修改，风险仍存在。
- dcp_size 应来自 ps.dcp_size 而非 server_args (design): 未采纳，当前仍使用 server_args.dcp_size。

风险与影响

- 风险：
  1. Speculative worker 挂起风险：五个 speculative draft worker 使用 replace(ps, pp_rank=0) 而未设置 pp_size=1，当目标 worker 配置了流水线并行（pp_size > 1）时

，draft worker 可能错误初始化流水线组，导致未定义行为或挂起。

2. 状态不一致：dcp_size 从 server_args 而非 ps.dcp_size 赋值，若 ps 的 dcp_size 与 server_args 不同（理论上不应），则存在读取不一致。当前代码中 ps 的 dcp_size 是在外部构造的，但从 server_args 直接取值更直接，但 reviewer 认为从 ps 读取更统一。
3. 回归影响：28 个文件的改动涉及核心路径，若遗漏某个调用点未更新参数，会导致运行时错误。由于参数签名变化，Python 会在运行时抛出 TypeError，可快速定位。 - 影响：影响范围：所有使用 ModelRunner、TpModelWorker 和 speculative worker 的地方，包括分布式初始化和基准测试。影响程度：中等。虽然改动文件多，但变更模式机械（参数替换），不改变运行时语义。所有内部字段名（self.tp_rank 等）保留，外部代码不受影响。需要团队在合并后对 speculative 流水线和 benchmark 进行回归测试。 - 风险标记：speculative worker 可能错误设置 pp_size，核心路径变更，benchmark 构造可能遗漏，review 问题未解决

关联脉络

- PR #31165 Drop ModelRunner's duplicated parallel-degree fields and read them via self.ps: 与本次 PR 同属 model_runner 重构系列，本 PR 引入 ps 对象，后续 PR 利用该对象统一内部状态访问。
- PR #31166 Narrow component dependencies to injected fields instead of ModelRunner: 本 PR 引入 ps 后，后续 PR 可以基于此进一步解耦组件对 ModelRunner 的依赖。
- PR #31163 Extract per-architecture KV-cache pool builders into KVCacheConfigurator: 同样属于 model_runner 重构系列，本 PR 提供的 ps 对象为后续重构提供统一的状态容器。