PR #24667 完整报告

sgl-project/sglang

feat: add SGLANG_RAY_BUNDLE_INDICES for fine-grained Ray bundle index control

合并时间：2026-05-30 17:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24667

执行摘要

- 一句话：新增 SGLANG_RAY_BUNDLE_INDICES 实现细粒度 Ray 工作者放置
- 推荐动作：建议仔细阅读 _resolve_bundle_indices 的验证逻辑和 _launch_scheduler_processes 的分支设计。Custom PG 模式下每个 bundle 只能包含 1 GPU 的假设被硬编码在 _validate_custom_placement_group 中，若未来需要支持多 GPU per bundle 需重新设计。该 PR 的设计模式（两分支分离、统一 actor 创建函数）值得在后续 Ray Engine 扩展中借鉴。对 DP 场景的 rank0_node_ip 修复也是关键改动。

功能与动机

Issue #24653 提出 SGLang 缺少类似 vLLM 的 VLLM_RAY_BUNDLE_INDICES 功能，导致用户无法在分布式部署中跳过故障 GPU、实现拓扑感知放置或集成 Ray Serve Gang Scheduling。现有自动创建的 placement group 每节点多 GPU，用户无法干预 bundle 分配。本 PR 通过新增 placement_group 参数和环境变量提供了精确控制能力。

实现拆解

1. 新增模块级辅助函数：在 python/sglang/srt/ray/engine.py 中添加 _get_bundle_node_ip, _compute_world_size, _resolve_bundle_indices, _validate_custom_placement_group, _create_scheduler_actor。这些函数分别负责查询指定 bundle 所在节点 IP、计算世界大小（GPU 总数）、解析环境变量中的 bundle 索引列表、验证自定义 placement group 结构、统一创建 SchedulerActor。
2. 修改 _launch_scheduler_processes 主路径：根据 server_args.placement_group 是否为 None 分流：
   - Auto PG 模式（默认）：保持原逻辑，自动创建 bundle 并查找 engine 所在 bundle，生成 bundle_for_node 列表。
   - Custom PG 模式（用户提供 placement_group）：验证 placement group 为每 bundle 1 GPU，通过 _resolve_bundle_indices 获取每个 rank 对应的 bundle 索引，并使用 _create_scheduler_actor 直接调度到指定 bundle。 其中 _validate_custom_placement_group 确保 bundle 数量不少于 world_size、每个 bundle 不超过 1 GPU，避免资源冲突。
3. 修改 RayDataParallelController：在 python/sglang/srt/ray/data_parallel_controller.py 中导入 engine 模块的辅助函数，重写 _launch_ray_tp_group 以支持 Custom PG 模式。

关键修复：每个 DP 组不再复用全局 rank0_node_ip，而是通过 _get_bundle_node_ip(pg, bundle_indices[start_rank]) 计算本组 rank-0 所在节点 IP，用于 dist_init_addr，避免多节点 DP 下 NCCL 初始化错误。

4. 声明环境变量：在 python/sglang/srt/environ.py 的 Envs 类中添加 SGLANG_RAY_BUNDLE_INDICES = EnvStr("")，使得 envs.SGLANG_RAY_BUNDLE_INDICES.get() 可被全局读取。
5. 测试和文档：在 test/manual/test_ray_engine.py 中新增 TestRayEnginePlacementGroup 类，包含五个测试用例覆盖基本功能和错误路径。错误测试类 TestRayEnginePlacementGroupErrors 验证验证失败场景。文档更新包括 offline_engine_api.ipynb 新增 Ray Integration 章节，以及 environment_variables.mdx 新增环境变量说明。

关键文件：

- python/sglang/srt/ray/engine.py（模块 Engine；类别 source；类型 dependency-wiring；符号 _get_bundle_node_ip, get_node_ip, _compute_world_size, _resolve_bundle_indices）：核心变更文件，新增 5 个辅助函数并重构 _launch_scheduler_processes 以支持 Auto/Custom PG 两模式。
- test/manual/test_ray_engine.py（模块 Ray Engine 测试；类别 test；类型 test-coverage；符号 TestRayEnginePlacementGroup, setUpClass, tearDownClass, test_custom_pg_dp1_tp2）：新增完整的 Custom PG 测试套件，包括正常路径和错误路径。
- python/sglang/srt/ray/data_parallel_controller.py（模块 DP 控制器；类别 source；类型 entrypoint）：导入 engine 模块辅助函数，重构 _launch_ray_tp_group 以支持 Custom PG，修复 DP rank0_node_ip 计算。
- python/sglang/srt/environ.py（模块 环境变量；类别 source；类型 core-logic）：声明 SGLANG_RAY_BUNDLE_INDICES 环境变量，供 _resolve_bundle_indices 读取。
- python/sglang/srt/ray/http_server.py（模块 HTTP 服务器；类别 source；类型 core-logic）：初始化 server_args.placement_group 为 None，确保 HTTP server 路径不干扰。
- docs_new/docs/basic_usage/offline_engine_api.ipynb（模块 文档；类别 other；类型 entrypoint）：新增 Ray Integration 章节，包含自定义 placement group 和 bundle index 控制的示例。
- docs_new/docs/references/environment_variables.mdx（模块 文档；类别 other；类型 core-logic）：记录新环境变量 SGLANG_RAY_BUNDLE_INDICES 的用途和格式。

关键符号：_get_bundle_node_ip, _compute_world_size, _resolve_bundle_indices, _validate_custom_placement_group, _create_scheduler_actor, RayEngine._launch_scheduler_processes, RayDataParallelController._launch_ray_tp_group

关键源码片段

python/sglang/srt/ray/engine.py

核心变更文件，新增 5 个辅助函数并重构 _launch_scheduler_processes 以支持 Auto/Custom PG 两模式。

```
# python/sglang/srt/ray/engine.py (head)

def _resolve_bundle_indices(pg: PlacementGroup, world_size: int) -> List[int]:
    """解析 bundle 索引列表，优先从环境变量读取，否则使用顺序索引。

    解析 `SGLANG_RAY_BUNDLE_INDICES` 环境变量（逗号分隔），
    验证长度与 world_size 一致、无重复、索引不越界。

    Args:
        pg: Placement group，用于获取 bundle 总数。
        world_size: 期望的索引个数（通过 _compute_world_size 预计算）。

    Returns:
        长度为 world_size 的 bundle 索引列表。
    """
    total_bundles = len(pg.bundle_specs)
    indices_str = envs.SGLANG_RAY_BUNDLE_INDICES.get()
    if not indices_str:
        # 未设置环境变量时，默认使用连续索引
        return list(range(world_size))

    indices = list(map(int, indices_str.split(",")))

    if len(indices) != world_size:
        raise ValueError(
            f"SGLANG_RAY_BUNDLE_INDICES has {len(indices)} values, "
            f"expected {world_size}"
        )

    if len(set(indices)) != len(indices):
        raise ValueError(f"SGLANG_RAY_BUNDLE_INDICES has duplicates: {indices}")

    for idx in indices:
        if idx < 0 or idx >= total_bundles:
            raise ValueError(f"Bundle index {idx} out of range [0, {total_bundles})")

    return indices
```

test/manual/test_ray_engine.py

新增完整的 Custom PG 测试套件，包括正常路径和错误路径。

```
# test/manual/test_ray_engine.py (head) —— 新增测试类

@unittest.skipUnless(_has_ray, "ray is not installed")
@unittest.skipUnless(_NUM_GPUS >= 2, "requires at least 2 GPUs")
class TestRayEnginePlacementGroup(unittest.TestCase):
```

```
"""测试 RayEngine 在自定义 placement_group 和 SGLANG_RAY_BUNDLE_INDICES
下的行为。"""

@classmethod
def setUpClass(cls):
    if not ray.is_initialized():
        ray.init(log_to_driver=True, runtime_env=_RAY_RUNTIME_ENV)

@classmethod
def tearDownClass(cls):
    ray.shutdown()

def test_custom_pg_dp1_tp2(self):
    """自定义 placement_group，dp_size=1, tp_size=2。"""
    from sglang.srt.ray.engine import RayEngine

    # 创建 2 个 bundle（各 1 GPU）的 placement group
    pg = placement_group([{"GPU": 1}] * 2, strategy="STRICT_PACK")
    ray.get(pg.ready())

    engine = RayEngine(
        model_path=_MODEL,
        tp_size=2,
        placement_group=pg,
        use_ray=True,
    )

    result = engine.generate("The capital of France is", _SAMPLING_PARAMS)
    self.assertIn("text", result)
    self.assertGreater(len(result["text"]), 0)
    print(f"Generated (dp=1, tp=2, custom PG): {result['text'][:200]}")

    engine.shutdown()
    ray.util.remove_placement_group(pg)

def test_bundle_indices_dp1_tp2(self):
    """通过 SGLANG_RAY_BUNDLE_INDICES 指定 bundle 顺序。"""
    from sglang.srt.ray.engine import RayEngine

    os.environ["SGLANG_RAY_BUNDLE_INDICES"] = "0,1"
    try:
        pg = placement_group([{"GPU": 1}] * 2, strategy="STRICT_PACK")
        ray.get(pg.ready())

        engine = RayEngine(
            model_path=_MODEL,
            tp_size=2,
            placement_group=pg,
            use_ray=True,
```

```
            )
            result = engine.generate("The capital of France is", _SAMPLING_PARAMS)
            self.assertIn("text", result)
            self.assertGreater(len(result["text"]), 0)
            engine.shutdown()
            ray.util.remove_placement_group(pg)
        finally:
            del os.environ["SGLANG_RAY_BUNDLE_INDICES"]
```

评论区精华

Review 中主要讨论点包括：

- placement_group 参数放置位置：xyuzh 和 Qiaolin-Yu 要求避免在 server_args.py 中引入 Ray 相关类型，最后改为由 RayEngine.__init__ 在构造 ServerArgs 后动态设置 placement_group 属性，通过 kwargs 传递。
- Custom PG 模式检测方式：最初使用 is_custom_pg 显式标志，但讨论后认为可通过 server_args.placement_group is not None 推断，因而移除该标志。
- DP 模式下 rank0_node_ip 计算错误：xyuzh 指出原实现中所有 DP 组均使用全局 rank-0 bundle 的节点 IP，导致跨节点 DP 组 NCCL 初始化失败。KaisennHu 修复为每个 DP 组独立解析本组 rank-0 bundle 所在 IP。
- 异常处理和验证缺失：xyuzh 多次要求补充异常处理和边界验证，最终添加了 _validate_custom_placement_group 并在环境变量解析时检查重复和越界。
- 代码重复与合并循环：reviewer 建议将两个独立循环合并，作者采纳。
  - placement_group 参数放置位置 (design): 不在 server_args 中添加静态字段，改由 RayEngine 动态赋值。
  - Custom PG 模式检测方式 (design): 移除 is_custom_pg 参数，用 server_args.placement_group 是否为 None 判断。
  - DP 下 rank0_node_ip 计算错误 (correctness): 每个 DP 组独立解析 rank0 节点 IP。
  - 缺少异常处理和验证 (correctness): 添加了完整的验证函数和异常抛出。
  - 测试覆盖 DP attention 场景 (testing): 已添加 DP attention 测试。
  - 重复代码和循环合并 (style): 合并循环，减少重复。
  - dp_server_args 重建后丢失 placement_group (correctness): 在 dataclasses.replace 后手动追加 placement_group。

风险与影响

- 风险：
  1. 回归风险：Auto PG 模式保持原逻辑，但新增代码分支（Custom PG）可能通过共享函数影响原路径。_compute_world_size 替换了原有的内联计算，需确认一致性。
  2. DP 场景仍存在隐患：尽管已修复 rank0_node_ip，但每个 DP 组独立解析 IP 引入了额外的 Ray remote 调用，可能引入延迟或超时。

3. 环境变量错误输入：SGLANG_RAY_BUNDLE_INDICES 若格式错误（非数字、重复、超范围），会抛出 ValueError。虽然已增加验证，但上游脚本未处理异常可能导致启动失败。
4. 安全性：placement_group 由用户传入，_validate_custom_placement_group 仅检查 GPU 数量，未对 bundle_specs 中的其他资源（如 CPU、内存）做限制，可能存在资源争抢。
5. 测试覆盖不足：错误测试仅覆盖了验证失败路径，未测试 Custom PG 下 Actor 异常、网络分区等场景。

- 影响：影响范围：
  - 用户：使用 Ray 运行时的用户可以通过 placement_group 参数和 SGLANG_RAY_BUNDLE_INDICES 环境变量精确控制 GPU bundle 选择。原有未设置的用户行为不变。
  - 系统：RayEngine._launch_scheduler_processes 和 RayDataParallelController._launch_ray_tp_group 逻辑重构，但通过分支保留了旧行为。DP 场景下 rank0_node_ip 计算方式改变，可能影响 NCCL 初始化 IP 地址。
  - 团队：引入新的辅助函数需被后续维护者理解；environ.py 新增环境变量需同步文档。

影响程度：中等。新功能默认关闭（不设置环境变量），不会对现有部署造成兼容性破坏。但 Custom PG 模式改变了 worker 调度方式，使用该模式的用户需理解 bundle 索引与 rank 的对应关系。

- 风险标记：核心路径变更，新增环境变量输入依赖，DP 场景修复关键，测试覆盖有限，未处理 Actor 异常

关联脉络

- 暂无明显关联 PR