

PR #2006 完整报告

radixark/miles

Launch inference engines through the inference engine worker spec

合并时间: 2026-09-04 13:48

原文链接: <http://prhub.com.cn/radixark/miles/pull/2006>

执行摘要

- 一句话: 实现推理引擎工作器规范, 统一推理引擎启动逻辑。
- 推荐动作: 本 PR 是重构系列的一部分, 展示了如何通过规范模式封装启动逻辑, 值得精读以学习设计模式。对于维护推理引擎启动代码的工程师, 建议重点关注 `_compute_spec_inference_engine` 和 `ServerCell` 的变更。

功能与动机

作为 Issue #1837 重构跟踪 issue 的一部分, 旨在将推理引擎的启动逻辑集中到规范 (spec) 中, 实现职责分离和代码复用, 以便后续支持更复杂的部署场景。

实现拆解

1. 扩展 `LaunchCommandContext`: 在 `miles/utils/workers/worker_spec.py` 中, 为 `LaunchCommandContext` 增加 `worker_in_cell_index` 和 `gpu_ids` 字段, 以支持推理引擎启动时所需的上下文信息。
2. 实现推理引擎规范: 在 `miles/ray/specs/inference.py` 中, 实现 `_compute_spec_inference_engine` 函数, 为每个服务器组配置生成 `CommandWorkerSpec`, 包含端口定义、环境变量和启动命令。同时, `specs_inference_engine` 函数现在返回这些规范的列表 (之前返回 `None`)。
3. 调整 `compute_engine_launch_cmd` 接口: 在 `miles/backends/sglang_utils/sglang_engine.py` 中, 将 `compute_engine_launch_cmd` 函数的参数从接受一个字典 `addr_and_ports` 改为接受多个具体参数 (`dist_init_addr`、`nccl_port` 等), 提高清晰度和类型安全性。
4. 更新 `ServerCell` 启动逻辑: 在 `miles/ray/rollout/server_cell.py` 中, `ServerCell` 增加 `model_idx`、`server_group_config` 和 `cell_index` 字段。`start_engines` 方法使用 `_compute_spec_inference_engine` 生成规范, 然后使用规范的环境变量和启动命令来启动 Ray actor, 并通过 `_compute_self_addrs` 辅助函数转换地址格式。
5. 更新调用方: 在 `miles/ray/rollout/router_manager.py` 和 `miles/ray/rollout/rollout_server.py` 中, 更新对 `LaunchCommandContext` 的构造, 提供新字段 `worker_in_cell_index` 和 `gpu_ids`。
6. 更新测试: 在 `tests/fast/backends/sglang_utils/test_sglang_engine.py` 中, 更新测试用例以适配新的 `compute_engine_launch_cmd` 接口。

关键文件:

- miles/ray/specs/inference.py (模块 推理规格; 类别 source; 类型 core-logic; 符号 `_compute_spec_inference_engine`, `_compute_launch_command`) : 核心变更, 实现了推理引擎规范的生成函数 `_compute_spec_inference_engine`。
- miles/ray/rollout/server_cell.py (模块 服务器单元; 类别 source; 类型 core-logic; 符号 `_compute_self_addrs`) : 更新 `ServerCell` 以使用推理引擎规范启动引擎, 实现职责分离。
- miles/backends/sglang_utils/sglang_engine.py (模块 SGLang 引擎; 类别 source; 类型 core-logic) : 调整 `compute_engine_launch_cmd` 函数签名, 从接受字典改为具体参数。
- miles/utils/workers/worker_spec.py (模块 工作器规范; 类别 source; 类型 core-logic) : 扩展 `LaunchCommandContext` 以支持推理引擎启动所需的上下文。
- tests/fast/backends/sglang_utils/test_sglang_engine.py (模块 SGLang 引擎测试; 类别 test; 类型 test-coverage) : 更新测试以适配 `compute_engine_launch_cmd` 新接口。

关键符号: `_compute_spec_inference_engine`, `specs_inference_engine`, `compute_engine_launch_cmd`, `_compute_self_addrs`

关键源码片段

miles/ray/specs/inference.py

核心变更, 实现了推理引擎规范的生成函数 `_compute_spec_inference_engine`。

```
def _compute_spec_inference_engine(
    args,
    model_idx: int,
    server_group_config: ServerGroupConfig,
) -> CommandWorkerSpec:
    """为给定的服务器组配置生成推理引擎的 CommandWorkerSpec。"""
    def _compute_launch_command(ctx: LaunchCommandContext) -> str:
        # 从上下文中获取分布式初始化地址
        dist_init = ctx.self_addrs["dist_init"]
        # 调用 compute_engine_launch_cmd 生成启动命令字符串
        return compute_engine_launch_cmd(
            args=args,
            # 使用 worker_in_cell_index 作为节点排名
            node_rank=ctx.worker_in_cell_index,
            worker_type=server_group_config.worker_type,
            base_gpu_id=ctx.gpu_ids[0],
            sglang_overrides=server_group_config.overrides,
            num_gpus_per_engine=server_group_config.num_gpus_per_engine,
            dist_init_addr=f"{dist_init.host}:{dist_init.port}",
            nccl_port=ctx.self_addrs["nccl"].port,
            host=ctx.self_addrs["primary"].host,
            port=ctx.self_addrs["primary"].port,
            disaggregation_bootstrap_port=d.port if (d := ctx.self_addrs.get("disaggregation_bootstrap")) else None,
            engine_info_bootstrap_port=ctx.self_addrs["engine_info_bootstrap"].port,
        )

    # 计算推理引擎所需的环境变量
```

```

envs = compute_inference_engine_env_vars(args)
# 返回完整的 CommandWorkerSpec
return CommandWorkerSpec(
    name=f"inference-engine-{model_idx}",
    port_infos=[
        PortInfo(name="primary", static_port=8000, allow_dynamic=True),
        PortInfo(name="dist_init", static_port=9000, mode="master", allow_dynamic=True),
        PortInfo(name="nccl", static_port=10000, allow_dynamic=True),
        PortInfo(name="disaggregation_bootstrap", static_port=11000, allow_dynamic=True),
        PortInfo(name="engine_info_bootstrap", static_port=12000, allow_dynamic=True),
    ],
    env_var=lambda: envs,
    scheduling=SchedulingSpec(
        num_cells=server_group_config.num_gpus // server_group_config.num_gpus_per_
        engine,
        num_workers_per_cell=max(1, server_group_config.num_gpus_per_engine // args.num_
        gpus_per_node),
        num_gpus_per_worker=0.2,
    ),
    launch_command=_compute_launch_command,
)

```

miles/ray/rollout/server_cell.py

更新 `ServerCell` 以使用推理引擎规范启动引擎，实现职责分离。

```

async def start_engines(self, port_allocator: PortAllocator) -> None:
    # ... 前置断言省略 ...
    # 获取推理引擎规范
    spec = _compute_spec_inference_engine(
        self.args,
        model_idx=self.model_idx,
        server_group_config=self.server_group_config,
    )
    # 使用规范的环境变量启动 Ray actor
    actor_handles = [
        launch_sglang_ray_actor(
            env_vars=spec.env_var(),
            pg=self.pg,
            gpu_index=self.gpu_offset + local_index * num_gpu_per_engine,
        )
        for local_index in range(self.num_nodes)
    ]
    # ... 分配端口等省略 ...
    # 使用规范的启动命令生成每个节点的启动命令
    launch_cmds = {
        rank: spec.launch_command(
            LaunchCommandContext(
                cell_index=self.cell_index,
                worker_in_cell_index=local_index,
            )
        )
        for rank in range(self.num_nodes)
    }

```

```
        self_addrs=_compute_self_addrs(addr_and_ports[rank]),
        spec_addrs={},
        gpu_ids=self.engine_gpu_ids[local_index],
    )
)
for local_index, rank in enumerate(global_ranks)
}
# ... 启动 actor 省略 ...
```

评论区精华

本 PR 仅有一个审核评论：审核者 guapisolo 批准，并指出 "issue fixed before #2176"。没有其他讨论，因此无明显争议。

- 审核批准 (other): 无争议，直接批准。

风险与影响

- 风险：变更涉及推理引擎的启动方式，如果规范计算错误，可能导致引擎启动失败。但由于是重构，逻辑与原有实现保持一致，且测试覆盖了 `compute_engine_launch_cmd` 的核心逻辑，风险较低。此外，`_compute_self_addrs` 是临时辅助函数，未来可能移除。
- 影响：本 PR 影响所有使用推理引擎启动的代码，包括路由器、会话服务器和滚动服务器的启动流程。通过统一到规范模式，提高了代码的可维护性和可扩展性，但未改变功能行为，对用户无直接影响。
- 风险标记：核心路径变更，接口重构

关联脉络

- PR #1842 Split RolloutManager into InferenceController and RolloutExecutor: 同属重构系列，将 RolloutManager 拆分为 InferenceController 和 RolloutExecutor，为本 PR 的规范模式奠定基础。
- PR #1843 Wire up InferenceController and RolloutExecutor and make the controller API async: 同属重构系列，将 InferenceController 和 RolloutExecutor 连接并异步化，与本 PR 的规范启动逻辑紧密相关。
- PR #1844 Rename RolloutExecutor.generate to get for future async and multi trainer scenarios: 同属重构系列，重命名方法以适应异步和多训练场景，与本 PR 的架构调整一致。