PR #1910 完整报告

radixark/miles

Replace the model config shell scripts with python

合并时间：2026-08-09 18:51

原文链接：http://prhub.com.cn/radixark/miles/pull/1910

执行摘要

- 一句话：模型配置脚本由 shell 重写为 Python，引入统一加载器
- 推荐动作：值得精读，尤其关注 load_model_args 的单行折叠设计与环境变量语义保留策略，这两处体现了对 shell 行为边界的深入理解。建议与本 PR 后紧邻的 #1911（inline model args 引号修复）和 #2279（快照测试移出 CI）一起阅读，可以完整看到一次大规模脚本迁移从引入、打补丁到成本再平衡的演进闭环。

功能与动机

PR body 仅注明 Part of #1837，属于该追踪 issue 的系列重构。核心动因在 commit 说明中：shell 版本删除后没有真相来源证明重写是否忠实，因此先对全部模型脚本 argv 做快照；同时 shell 脚本难以测试、难以组合，且 read -ra ... <<< "$(...)" 遇到换行会静默截断，这些脆弱点只有换成 Python 才能在加载层统一处理。

实现拆解

1. 新增统一加载器：miles/utils/external_utils/model_args_utils.py 定义 load_model_args()，通过 importlib.util 按路径导入 scripts/models/.py，调用其 model_args(**kwargs) 并把结果折叠成单行（换行会被 shell read -ra 静默截断）。模块名对点号和短横线做替换，支持 glm4.5-106B-A12B 这类无法直接 import 的文件名。
2. 迁移 62 个模型脚本：scripts/models/*.sh 全部重写为 .py（如 deepseek-v4-flash.py、deepseek-v4-pro.py、moonlight.py、deepseek-v32.py、deepseek-v3.py、joyai-llm-flash.py、kimi-k2.py、qwen3-next-80B-A3B.py、qwen3.5/qwen3.6 系列、glm4.7-flash.py、glm5-744B-A40B.py 等）。每个模块声明 model_args()，用模型脚本可调用的 moe_layer_freq() 生成 --moe-layer-freq 掩码；环境变量覆盖语义与 shell 时代对齐，统一采用 x if x is not None else os.environ.get(...) or default，既保留 0 值真值语义，也保留空字符串回退默认。
3. 改造消费方：launcher 与命令构建从 source .sh 改为调用 load_model_args；shell 侧保留 read -ra 兼容，通过 python model_args_utils.py <model_type> 命令行读出 token 流；NPU docker patch 同步指向 Python 模型定义；p2p profile 的 rotary_base 覆盖在 16 节点 launcher 中保留传递。
4. 测试与快照配套：tests/fast/utils/external_utils/test_model_args_utils.py 覆盖拆分语义、跨行折叠、环境变量优先级、0 值、拼写 TypeError、兄弟脚本加载以及 bash 子进程兼容；迁移前已生成 tests/snapshots 下全部模型 argv 快照，迁移只允许改 producer 不允许

改 golden 文件；run_megatron CLI 测试改指向加载器。

关键文件：

- miles/utils/external_utils/model_args_utils.py（模块 参数加载；类别 source；类型 data-contract；符号 load_model_args, load_sibling_model_args, moe_layer_freq, import_module_from_path）：新增的统一加载器，是整个迁移的契约中心：负责按路径导入模型脚本、折叠换行、暴露 moe_layer_freq 与 load_sibling_model_args。
- tests/fast/utils/external_utils/test_model_args_utils.py（模块 参数加载；类别 test；类型 test-coverage；符号 model_args, model_script, TestLoadModelArgsSplitting, test_splits_each_line_the_way_read_ra_would）：233 行新测试锁定加载器契约：拆分语义、环境变量优先级、0 值、兄弟加载、bash 子进程兼容，是迁移忠实性的证明。
- scripts/models/deepseek-v4-flash.py（模块 模型配置；类别 source；类型 data-contract；符号 model_args）：代表 62 个被重写的模型脚本：展示环境变量覆盖、MoE 掩码生成、常量注入，以及模型插件 spec 的声明方式。
- scripts/models/deepseek-v4-pro.py（模块 模型配置；类别 source；类型 data-contract；符号 model_args）：与 flash 同族的 Pro 规格迁移范本，同样展示环境覆盖、MoE 掩码与常量注入；代表同一批迁移文件的重复模式。

关键符号：load_model_args, load_sibling_model_args, moe_layer_freq, import_module_from_path, model_args

关键源码片段

miles/utils/external_utils/model_args_utils.py

新增的统一加载器，是整个迁移的契约中心：负责按路径导入模型脚本、折叠换行、暴露 moe_layer_freq 与 load_sibling_model_args。

```
# miles/utils/external_utils/model_args_utils.py
# 模型参数脚本的 Python 加载器。
# shell 时代用 source 执行 scripts/models/*.sh；现在改为导入 .py 并调用 model_args()。

import importlib.util
import sys
from pathlib import Path
from types import ModuleType

REPO_ROOT = Path(__file__).resolve().parents[3]
MODEL_SCRIPT_DIR = REPO_ROOT / "scripts" / "models"


def load_model_args(model_type: str, model_script_dir: Path | None = None, **kwargs: object) -
> str:
    # 把脚本折叠成一行：shell 的 `read -ra ... <<< "$(...)"` 遇到换行会静默截断，
    # 返回值必须保持单行 token 流。
    path = (model_script_dir or MODEL_SCRIPT_DIR) / f"{model_type}.py"
    assert path.exists(), f"no model args script at {path}"
    # 让被加载的脚本可以 `from model_args_utils import ...`，即使 miles 未安装
```

```
    sys.modules.setdefault("model_args_utils", sys.modules[__name__])
    # 模型名可能含点和短横线（如 glm4.5-106B-A12B），不能直接作为模块名
    module_name = f"miles_model_args_{path.stem.replace('.', '_').replace('-', '_')}"
    module = import_module_from_path(path, module_name)
    args = " ".join(module.model_args(**kwargs).split())
    assert args, f"{path} declared no model args" # 全空声明是生成器 bug，而不是合法的空参数
    return args


def load_sibling_model_args(model_script: str, model_type: str, **kwargs: object) -> str:
    # 变体脚本（如 qwen3.5-35B-A3B_lora）必须加载自己 checkout 里的父模型，
    # 而不是已安装包里的同名文件。
    return load_model_args(model_type, model_script_dir=Path(model_script).resolve().parent,
    **kwargs)


def moe_layer_freq(*, nlayers: int, first_k_dense_replace: int) -> str:
    # 渲染 Megatron 的 --moe-layer-freq：前 K 层 dense，其余 MoE。
    # 注意 min()：shell 时代的循环按层数跑，2 层 deepseek-v3 得到 [0,0] 而不是 [0,0,0]。
    dense = min(first_k_dense_replace, nlayers)
    return "[" + ",".join(["0"] * dense + ["1"] * (nlayers - dense)) + "]"


def import_module_from_path(path: Path, module_name: str) -> ModuleType:
    spec = importlib.util.spec_from_file_location(module_name, path)
    assert spec is not None and spec.loader is not None, f"cannot load {path}"
    module = importlib.util.module_from_spec(spec)
    sys.modules[module_name] = module
    try:
        spec.loader.exec_module(module)
    finally:
        # 导入完成后清掉临时模块名，避免污染 sys.modules
        del sys.modules[module_name]
    return module
```

tests/fast/utils/external_utils/test_model_args_utils.py

233 行新测试锁定加载器契约：拆分语义、环境变量优先级、0 值、兄弟加载、bash 子进程兼容，是迁移忠实性的证明。

```
# tests/fast/utils/external_utils/test_model_args_utils.py
# 把加载器的行为契约锁死在测试里，迁移 shell 时才不会悄悄改变语义。

@pytest.fixture

def model_script(monkeypatch, tmp_path):
    # 用假模型脚本代替真实 scripts/models 目录，避免测试依赖真实模型参数
    path = tmp_path / "fake-model.4layer.py"
    path.write_text(_SCRIPT_BODY)
    (tmp_path / "fake-wrapper.py").write_text(_WRAPPER_BODY)
```

```
    monkeypatch.setattr("miles.utils.external_utils.model_args_utils.MODEL_SCRIPT_DIR", tmp_
    path)
    monkeypatch.delenv("MODEL_ARGS_NUM_LAYERS", raising=False)
    return path


class TestLoadModelArgsSplitting:
    def test_collapses_a_declaration_that_spans_several_lines(self, model_script):
        # 换行会让 launcher 的 `read -ra ... <<< "$(...)"` 在第一行后静默截断，
        # 所以加载器必须把返回值折叠成单行。
        model_script.write_text("def model_args() -> str:\n    return '--a 1\
--b 2'\n")
        assert load_model_args("fake-model.4layer") == "--a 1 --b 2"

    def test_keeps_the_bracket_patterns_megatron_expects(self, model_script):
        # --moe-layer-freq 包含方括号和星号，必须作为单个 token 穿过 read -ra
        model_script.write_text("def model_args() -> str:\n    return '--moe-layer-freq [0]*3+[1]
        *75'\n")
        assert load_model_args("fake-model.4layer") == "--moe-layer-freq [0]*3+[1]*75"


def test_shell_consumers_recover_the_original_tokens():
    # shell 消费端用 read -ra 读回 token，必须与原 argv 一致；
    # 这个子进程测试验证的是 bash 与 Python 加载器之间的桥接契约。
    script = (
        f'set -e; MODEL_ARGS_LINE="$({sys.executable} {_MODEL_ARGS_CLI} qwen3-4B)" || exit
        1; '
        'read -ra MODEL_ARGS <<< "${MODEL_ARGS_LINE}"; printf "%s\\n" "${MODEL_ARGS[@]}
        "'
    )
    result = subprocess.run(["bash", "-c", script], capture_output=True, text=True, check=True)
    assert result.stdout.splitlines() == load_model_args("qwen3-4B").split()
```

scripts/models/deepseek-v4-flash.py

代表 62 个被重写的模型脚本：展示环境变量覆盖、MoE 掩码生成、常量注入，以及模型插件 spec 的声明方式。

```
# scripts/models/deepseek-v4-flash.py
# 从 shell 时代迁移来的模型参数定义：一个 model_args() 返回 megatron 命令行 flag。
import os

from model_args_utils import moe_layer_freq


COMPRESS_RATIOS = "0 0 4 128 4 128 4 128 4 128 4 128 4 128 4 128 4 128 4 128 4 128 4
128 4 128 4 128 4 128 4 128 4 128 4 128 4 128 4 128 4 0"
SWIGLU_LIMIT_ARGS = "--activation-func-clamp-value 10 --no-bias-swiglu-fusion --no-activation-
func-clamp-shared-expert"
```

```
def model_args(
    nlayers: int | None = None, rotary_scaling_factor: str | None = None, compress_ratios: str =
    COMPRESS_RATIOS
) -> str:
    # 用 None 判断而不是 `or`：0 是合法层数，空字符串要回退默认值
    nlayers = nlayers if nlayers is not None else int(os.environ.get("MODEL_ARGS_NUM_
    LAYERS") or 43)
    rotary_scaling_factor = (
        rotary_scaling_factor if rotary_scaling_factor is not None else os.environ.get("ROTARY_
        SCALING_FACTOR") or "16"
    )
    return (
        "--disable-bias-linear "
        f"--num-layers {nlayers} "
        "--hidden-size 4096 "
        "--ffn-hidden-size 2048 "
        "--num-attention-heads 64 "
        "--normalization RMSNorm "
        "--position-embedding-type rope "
        "--norm-epsilon 1e-6 "
        "--swiglu "
        "--untie-embeddings-and-output-weights "
        "--vocab-size 129280 "
        "--hidden-dropout 0.0 "
        "--attention-dropout 0.0 "
        # MLA 参数（V4 复用）
        "--multi-latent-attention "
        "--q-lora-rank 1024 "
        "--kv-lora-rank 512 "
        "--qk-head-dim 512 "
        "--qk-pos-emb-head-dim 64 "
        "--v-head-dim 512 "
        "--qk-layernorm "
        f"--rotary-scaling-factor {rotary_scaling_factor} "
        "--rotary-base 10000 "
        "--original-max-position-embeddings 65536 "
        "--beta-fast 32 "
        "--beta-slow 1 "
        "--attention-softmax-in-fp32 "
        "--no-rope-fusion "
        # MoE
        "--num-experts 256 "
        # DeepSeek V4 没有 dense 前缀，所以 first_k_dense_replace=0 生成全 1 掩码
        f"--moe-layer-freq {moe_layer_freq(nlayers=nlayers, first_k_dense_replace=0)} "
        "--moe-ffn-hidden-size 2048 "
        "--moe-router-topk 6 "
        "--moe-shared-expert-intermediate-size 2048 "
        "--moe-router-pre-softmax "
        "--moe-router-score-function sqrtsoftplus "
```

```
        "--moe-router-enable-expert-bias "
        "--moe-router-load-balancing-type seq_aux_loss "
        "--moe-token-dispatcher-type alltoall "
        "--moe-aux-loss-coeff 0 "
        "--moe-grouped-gemm "
        "--moe-router-topk-scaling-factor 1.5 "
        # DSV4 专属参数
        "--experimental-attention-variant dsv4 "
        "--dsv4-hc-mult 4 "
        "--dsv4-hc-sinkhorn-iters 20 "
        f"--dsv4-compress-ratios {compress_ratios} "
        "--dsv4-compress-rope-theta 160000 "
        "--dsv4-o-groups 8 "
        "--dsv4-o-lora-rank 1024 "
        "--dsv4-n-hash-layers 3 "
        "--dsv4-window-size 128 "
        # DSA Indexer
        "--dsa-indexer-n-heads 64 "
        "--dsa-indexer-head-dim 128 "
        "--dsa-indexer-topk 512 "
        # V4 模型插件 spec
        "--spec miles_plugins.models.deepseek_v4.deepseek_v4 get_dsv4_spec "
        f"{SWIGLU_LIMIT_ARGS} "
    )
```

评论区精华

仓库未留下 review 评论，唯一审核人 yueming-yuan 直接 APPROVED，未附说明。设计权衡主要体现在 commit 说明中：

> "Once the shell versions are gone there is no source of truth left to prove the rewrite was faithful, so record the argv each of them expands to now." "A newline would truncate the shell's read -ra." 这两条分别说明了快照先行策略与单行折叠的必要性，是本 PR 最值得注意的两个设计决策。

- model_args() 必须折叠成单行：read -ra 静默截断 (correctness): load_model_args 用 join().split() 折叠空白并断言非空，测试锁定拆分语义。
- 环境变量覆盖语义保留：0 值与空字符串 (design): 统一用 x if x is not None else os.environ.get(...) or default 模式，测试覆盖 0、空串、优先级。
- 方括号 token 与后续 shell 引号问题 (correctness): 本 PR 在加载器与测试中保护 token 完整性，#1911 补上 shell 引号。
- 先快照后迁移的忠实性证明 (testing): 迁移只允许改 producer 不允许改 golden 文件；后续 #2279 将该快照测试移出 CI。

风险与影响

- 风险：

1. 批量重写回归风险：62 个模型脚本全部重写，虽然快照测试锁定了 argv，但 shell 语义与 Python 语义仍可能有细微差异，尤其是内联参数被 shell 重新解析时的引号 /glob 展开问题，后续 PR #1911 证实了该风险并修复。
2. 删除 .sh 属 breaking change：scripts/models/*.sh 被移除，外部用户若仍 source 这些脚本会直接失败；NPU docker patch 与 docs 虽已同步，但灰度期需留意兼容。
3. 导入副作用：import_module_from_path 向 sys.modules 临时注册模块名并在 finally 中删除；同时通过 sys.modules.setdefault 暴露 model_args_utils 别名，依赖全局解释器状态，在并发导入场景下有理论冲突风险，但模块名按 stem 区分，实际风险低。
4. 测试依赖 bash 子进程：测试用 subprocess 跑 bash 验证桥接契约，跨平台性受限，且快照测试体量大，后续 #2279 已将其移出 CI 改手动执行。 - 影响：影响范围覆盖训练启动全链路：所有模型参数定义从 shell 变为 Python，新增模型只需写一个声明 model_args() 的 .py；launcher 与 CLI 的模型参数获取路径统一收敛到 load_model_args；测试体系获得可组合、可参数化的模型配置，同时新增快照回归防护。对团队而言，脚本可测试性、可读性与可维护性显著提升；对外部用户是行为兼容的 breaking change（.sh 消失），需要随文档与 docker patch 一起升级。 - 风险标记：62 个模型脚本批量重写，删除 .sh 为 breaking change，shell 引号兼容风险，快照测试依赖 bash 环境

关联脉络

- PR #1911 Quote the model args miles inlines into the launch command: 同一 #1837 链条的紧邻后续 PR：#1910 把 model args 内联进 launch 命令后，shell 引号 /glob 展开问题由 #1911 修复，并同步更新 model_args_utils 相关测试与 launch 脚本快照。
- PR #2279 Run the launch script snapshot tests by hand instead of in CI: #1910 建立的快照测试体系后续被认为不适合常驻 CI，改由手动执行，反映该测试成本与收益的再平衡。