PR #2353 完整报告

radixark/miles

dashboard: report model FLOPs utilization

合并时间：2026-08-13 01:47

原文链接：http://prhub.com.cn/radixark/miles/pull/2353

执行摘要

- 一句话：按设备峰值计算 MFU，修复 FSDP 缺失吞吐指标并上线看板
- 推荐动作：值得精读。该 PR 的工程判断密度高：分子分母同落盘保证指标可审计、保持 train-step 口径避免异步误导、未知设备“不猜默认值”、阈值从常量演进为可配置 flag，都是可复用的可观测性设计模式。重点看 device_flops.py 的整词匹配实现、flops_args_from_hf_config() 的容错层级，以及 review 中三次“实测数据推翻默认假设”的往复——它们展示了如何用真实运行数据校准一个看似合理的常量。

功能与动机

PR body 明确点出核心缺口：“perf/actor_train_tflops already measures per-GPU throughput. MFU is that divided by the device's peak — and nothing in miles knew what any device's peak was, so the ratio could not be formed.” 同时：“the FSDP backend passed compute_total_fwd_flops=None, so FSDP runs had nothroughput metrics at all.” 即已有吞吐指标但缺分母、FSDP 运行连吞吐指标都没有。此外 PR 还希望区分“训练步算得慢”与“训练步等 rollout 数据”两种瓶颈——文中给出 colocate 与 fully-async 下 wait_time_ratio 从 0.708 塌缩到 0.060 的对比，说明 MFU 应保持为 train-step 比率而非端到端混合数，避免“async 让训练高效 3.5 倍”的错误结论。

实现拆解

1. 新增设备峰值算力表（分母来源）：新建 miles/utils/device_flops.py，用 _PEAK_BF16_TFLOPS 字典登记 A100（312）、H100（989）、H100 PCIE（756）、H200（989）、GH200（989）、B200（2250）、B300（2250）、GB200/GB300（2500，单位 TFLOP/s）的 dense BF16 峰值。_words() 归一化设备名字符与大写，_MATCH_ORDER 按 key 长度降序做整词匹配，避免 "NVIDIA H100-PCIE-80GB" 这类连字符写法落空；_current_device_name() 用 @cache 缓存 torch.cuda.get_device_name() 结果，local_peak_bf16_tflops() 返回本机峰值，未知设备返回 None 而不是猜默认值。配套 --mfu-peak-tflops 参数（miles/utils/arguments.py）允许用户覆盖表内数值。
2. 打通 HF 配置与 FLOPs 模型，修复 FSDP 无吞吐指标：miles/utils/flops_utils.py 新增 fwd_tflops_per_gpu()，把两后端此前各自内联的 / world_size / 1e12 约定收敛为单一函数；新增 flops_args_from_hf_config() 将 HF PreTrainedConfig 翻译成 calculate_fwd_flops() 所需的 SimpleNamespace，覆盖 GQA 的 num_key_value_heads、

MoE 的 n_routed_experts / num_local_experts、DeepSeek 的 first_k_dense_replace / n_shared_experts、Qwen 的 decoder_sparse_step / mlp_only_layers 混合稀疏模式、MLA 维度，以及多模态 get_text_config() 文本塔解包。miles/backends/fsdp_utils/actor.py 由传 compute_total_fwd_flops=None 改为传真实函数，FSDP 运行首次获得与 megatron 同口径的 FLOPs 与吞吐指标。

3. 指标发射与可审计性：miles/utils/train_metric_utils.py 的 log_perf_data_raw() 在已有 perf/actor_train_tflops 基础上新增 perf/actor_train_mfu（吞吐 / 设备峰值）与 perf/mfu_peak_tflops——分母随分子一起落盘，读指标的人可复算；设备未识别或没有 FLOPs 模型时，MFU 字段整体省略而非对假定峰值输出百分比。零训练时间、非主 rank 等边界照旧不产出指标。
4. Dashboard 展示与告警：miles/dashboard/advisory.py 新增 mfu_summary()（排除 step 0，汇总 latest / mean / steps / peak）与 _mfu_advisories()（稳态 step ≥ 3 且均值低于阈值时告警，消息说明"算得慢不是等数据"及其典型成因）；serve.py 新增 --low-mfu-threshold 启动参数（0 关闭规则），server.py 的 /api/advisory 同时返回 mfu summary 与 advisories，保证前端 tile 与告警规则使用同一份均值口径。前端 views_timeline.js 新增 renderMfu() tile（latest、mean over steps、device peak 三个 statBox），app.js 提取公共 statBox() 组件供复用。
5. 测试与文档配套：新增 tests/fast/utils/test_device_flops.py（整词匹配、大小写、PCIe 特例、GH200 不误读为 H200、未知设备返回 None）、tests/fast/utils/test_flops_from_hf_config.py（HF 与 megatron 参数在 dense / MoE / MLA / 多模态下的 FLOPs 一致性、专家宽度回退与 dense-prefix 钳制）、tests/fast/utils/test_train_metric_utils.py（MFU = 吞吐 / 峰值、分母可复算、未知峰值省略 MFU、无 FLOPs 模型不产出）；扩充 tests/fast/dashboard/test_advisory.py（阈值可配置、0 禁用、step 0 排除、最少步数、无指标不告警）与 tests/fast/dashboard/test_server.py（端点同服务 MFU 与分母、advisory 返回规则所用 summary）。docs/user-guide/dashboard.md 新增 "Model FLOPs utilization" 小节，说明公式、约定、注意事项与实测基线。

关键文件：

- miles/utils/device_flops.py（模块 设备算力；类别 source；类型 core-logic；符号 _words, peak_bf16_tflops, _current_device_name, local_peak_bf16_tflops）：新增的设备峰值算力表是整个 MFU 分母的来源：登记 9 款 GPU 的 dense BF16 峰值、整词匹配避免连字符设备名落空、未知设备返回 None 而非猜默认值，直接决定 MFU 数值的准确性与缺失行为。
- miles/utils/flops_utils.py（模块 FLOPs 计算；类别 source；类型 core-logic；符号 fwd_tflops_per_gpu, _first, _moe_layer_pattern, flops_args_from_hf_config）：新增 flops_args_from_hf_config() 把 HF 配置翻译成 calculate_fwd_flops() 可读的 SimpleNamespace，是 FSDP 后端获得 FLOPs 指标的关键适配层；fwd_tflops_per_gpu() 统一了两后端的 / world_size / 1e12 约定。
- tests/fast/utils/test_flops_from_hf_config.py（模块 单元测试；类别 test；类型 test-coverage；符号 megatron_args, hf_config, assert_same, test_dense_gqa_model_matches）：核心 parity 测试：用同一组 SEQLENS 断言 HF 配置适配后的 FLOPs 与 megatron 参数完全一致，覆盖 dense/GQA、Mixtral/gpt-oss、

DeepSeek dense-prefix、Qwen sparse-step、MLA、多模态解包等全部适配分支，是防回归的主要屏障。

- miles/dashboard/advisory.py（模块 效率建议；类别 source；类型 core-logic；符号 compute_advisories, mfu_summary, _mfu_advisories）：新增 mfu_summary() 与 _mfu_advisories()：前者服务端统一计算均值（排除 step 0），后者在稳态 step ≥ 3 且均值低于阈值时产出告警；compute_advisories 增加 mfu / low_mfu 入参，与 serve 层共用同一 summary，保证 tile 与规则口径一致。
- miles/utils/train_metric_utils.py（模块 训练指标；类别 source；类型 dependency-wiring）：log_perf_data_raw() 在此发射 perf/actor_train_mfu 与 perf/mfu_peak_tflops，把设备峰值与 MFU 比值的记录逻辑落到训练侧，是整条指标链路的源头。
- tests/fast/utils/test_train_metric_utils.py（模块 单元测试；类别 test；类型 test-coverage；符号 timer, logged, make_args, run）：验证 MFU 数学关系（吞吐 / 峰值）、分母随分子落盘可审计、未知峰值省略 MFU、无 FLOPs 模型不产出、零训练时间与非主 rank 边界，把新指标的行为契约固定下来。
- tests/fast/utils/test_device_flops.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_known_devices_resolve, test_only_pcie_h100_clocks_below_its_family, test_grace_hopper_is_not_read_as_an_h200, test_longer_key_wins_over_the_substring_it_contains）：锁定设备名匹配的全部边界：已知设备解析、PCIe 特例、GH200 不误读为 H200、整词优先、大小写不敏感、未知设备返回 None、无 CUDA 返回 None，是峰值表准确性的守护测试。
- miles/dashboard/static/views_timeline.js（模块 前端视图；类别 source；类型 core-logic；符号 renderMfu, pct）：新增 MFU tile（renderMfu）与 UTIL_FILL 样式，接入 /api/advisory 返回的 mfu summary，是用户直接看到新指标的入口；statBox 组件同时落到 app.js 供复用。
- tests/fast/dashboard/test_advisory.py（模块 单元测试；类别 test；类型 test-coverage；符号 _mfu_store, test_sustained_low_mfu_is_a_warning, test_healthy_mfu_is_quiet, test_threshold_is_configurable）：覆盖低 MFU 告警的完整行为矩阵：持续低值告警、健康静默、阈值可配置、0 禁用、step 0 排除、稳态步数不足不判断、无 MFU 指标不告警。
- miles/dashboard/serve.py（模块 仪表盘服务；类别 source；类型 dependency-wiring）：把 DEFAULT_LOW_MFU 暴露为 --low-mfu-threshold 启动参数并传入 make_app，是 review 中“阈值交给用户”决策的落地点。
- miles/backends/fsdp_utils/actor.py（模块 FSDP 后端；类别 source；类型 dependency-wiring）：FSDP 后端此前传 compute_total_fwd_flops=None 导致完全没有吞吐指标，本次改为传真实 FLOPs 函数，是 FSDP 运行获得 MFU 的前提修复。
- docs/user-guide/dashboard.md（模块 用户文档；类别 docs；类型 documentation）：新增 Model FLOPs utilization 章节，沉淀公式、约定、注意事项（稀疏峰值、MTP 缺口、recompute 影响）与实测基线，是团队后续校准阈值的重要参考。

关键符号：peak_bf16_tflops, local_peak_bf16_tflops, fwd_tflops_per_gpu, flops_args_from_hf_config, _moe_layer_pattern, mfu_summary, _mfu_advisories, compute_advisories, renderMfu, log_perf_data_raw

关键源码片段

miles/utils/device_flops.py

新增的设备峰值算力表是整个 MFU 分母的来源：登记 9 款 GPU 的 dense BF16 峰值、整词匹配避免连字符设备名落空、未知设备返回 None 而非猜默认值，直接决定 MFU 数值的准确性与缺失行为。

```
# miles/utils/device_flops.py — 设备名 → 峰值算力映射，作为 MFU 的分母

from __future__ import annotations

import re
from functools import cache

# 各芯片峰值 dense BF16 吞吐（TFLOP/s）。数据手册常宣传 2:4 稀疏值，
# 恰好是 dense 的两倍，若引用会令全部 MFU 减半，因此这里统一使用 dense 值
_PEAK_BF16_TFLOPS: dict[str, float] = {
    "A100": 312.0,
    "H100": 989.0,
    "H100 PCIE": 756.0, # PCIe 版是 114-SM 阉割芯片；NVL 版为 132-SM 全尺寸，仍用 989.0
    "H200": 989.0,
    "GH200": 989.0,
    "B200": 2250.0,
    "B300": 2250.0, # Blackwell Ultra 的 FP8/NVFP4 提升明显，但 dense BF16 与 B200 持平
    "GB200": 2500.0,
    "GB300": 2500.0,
}

# 按 key 长度降序匹配，保证 "GB300" 优先于其子串 "B300"
_MATCH_ORDER: tuple[str, ...] = tuple(sorted(_PEAK_BF16_TFLOPS, key=len, reverse=True))


def _words(device_name: str) -> str:
    # 统一分隔符并大写，使 "H100-PCIE-80GB" 也能命中 "H100 PCIE" 键
    return f" {re.sub(r'[^A-Za-z0-9]+', ' ', device_name).upper().strip()} "


def peak_bf16_tflops(device_name: str) -> float | None:
    name = _words(device_name)
    for key in _MATCH_ORDER:
        if f" {key} " in name:
            return _PEAK_BF16_TFLOPS[key]
    return None # 未识别设备不猜测峰值，上层据此省略 MFU 而非输出错误百分比


@cache
# 惰性导入 torch，避免纯 CPU / 无 CUDA 环境被拖入依赖
# 返回值缓存整次进程，设备名在运行期内不会变化
```

```
def _current_device_name() -> str | None:
    import torch

    if not torch.cuda.is_available():
        return None
    return torch.cuda.get_device_name()


def local_peak_bf16_tflops() -> float | None:
    device_name = _current_device_name()
    return peak_bf16_tflops(device_name) if device_name else None
```

miles/utils/flops_utils.py

新增 flops_args_from_hf_config() 把 HF 配置翻译成 calculate_fwd_flops() 可读的 SimpleNamespace，是 FSDP 后端获得 FLOPs 指标的关键适配层；fwd_tflops_per_gpu() 统一了两后端的 / world_size / 1e12 约定。

```
# miles/utils/flops_utils.py — HF 配置 → megatron FLOPs 模型参数的适配层

def fwd_tflops_per_gpu(seqlens, args, world_size):
    # 收敛两后端此前各自内联的约定：总 FLOPs 除以世界大小再除以 1e12
    return calculate_fwd_flops(seqlens, args) / world_size / 1e12


def _first(config, *names, default=None):
    # HF 各系列字段命名不统一，依次尝试候选名，取到非 None 即返回
    for name in names:
        value = getattr(config, name, None)
        if value is not None:
            return value
    return default


def _moe_layer_pattern(config, num_layers, num_experts):
    # 生成逐层是否 MoE 的掩码：DeepSeek 风格用 first_k_dense_replace，
    # Qwen 风格用 decoder_sparse_step + mlp_only_layers
    if num_experts is None:
        return None
    first_dense = getattr(config, "first_k_dense_replace", None)
    if first_dense is not None:
        return [0 if i < first_dense else 1 for i in range(num_layers)]
    mlp_only = set(getattr(config, "mlp_only_layers", None) or ())
    step = getattr(config, "decoder_sparse_step", 1) or 1
    return [0 if i in mlp_only or (i + 1) % step != 0 else 1 for i in range(num_layers)]


def flops_args_from_hf_config(config):
    # 多模态配置先解开 text tower：按文本塔而不是视觉塔的尺寸估算 FLOPs
    getter = getattr(config, "get_text_config", None)
```

```
    config = (getter() if callable(getter) else getattr(config, "text_config", None)) or config

    num_attention_heads = config.num_attention_heads
    hidden_size = config.hidden_size
    num_layers = _first(config, "num_hidden_layers", "num_layers")
    assert num_layers is not None, f"no layer count on {type(config).__name__}; cannot size the
    FLOPs model"
    num_experts = _first(config, "n_routed_experts", "num_experts", "num_local_experts")

    shared_ffn = getattr(config, "shared_expert_intermediate_size", None)
    # DeepSeek 风格的 n_shared_experts 换算成等效中间层宽度
    if shared_ffn is None and getattr(config, "n_shared_experts", None):
        shared_ffn = config.n_shared_experts * config.moe_intermediate_size

    moe_ffn = getattr(config, "moe_intermediate_size", None)
    # Mixtral / gpt-oss 风格没有 moe_intermediate_size，直接用 intermediate_size；
    # 只有声明了专家数才回退，dense 配置保持无专家宽度
    if num_experts is not None and moe_ffn is None:
        moe_ffn = config.intermediate_size

    return SimpleNamespace(
        hidden_size=hidden_size,
        num_attention_heads=num_attention_heads,
        num_query_groups=_first(config, "num_key_value_heads", default=num_attention_heads),
        vocab_size=config.vocab_size,
        num_layers=num_layers,
        ffn_hidden_size=config.intermediate_size,
        kv_channels=_first(config, "head_dim", default=hidden_size // num_attention_heads),
        num_experts=num_experts,
        moe_ffn_hidden_size=moe_ffn,
        moe_router_topk=_first(config, "num_experts_per_tok", "moe_topk", default=1),
        moe_shared_expert_intermediate_size=shared_ffn,
        moe_layer_freq=_moe_layer_pattern(config, num_layers, num_experts),
        q_lora_rank=getattr(config, "q_lora_rank", None),
        kv_lora_rank=getattr(config, "kv_lora_rank", None),
        qk_head_dim=getattr(config, "qk_nope_head_dim", None) or 0,
        qk_pos_emb_head_dim=getattr(config, "qk_rope_head_dim", None) or 0,
        v_head_dim=getattr(config, "v_head_dim", None) or 0,
    )
```

miles/dashboard/advisory.py

新增 mfu_summary() 与 _mfu_advisories()：前者服务端统一计算均值（排除 step 0），后者在稳态 step ≥ 3 且均值低于阈值时产出告警；compute_advisories 增加 mfu / low_mfu 入参，与 serve 层共用同一 summary，保证 tile 与规则口径一致。

```
# miles/dashboard/advisory.py — MFU 汇总与低利用率告警（Efficiency 视图）

# 阈值默认 0.15 只是起点：健康运行的 MFU 因模型与配置跨度极大
#  （小 dense 模型约 25%，带 CPU offload 的 MoE 约 5%，长上下文 K2.5 约 17%），
```

```
# 绝对阈值无法一劳永逸，serve 层提供 --low-mfu-threshold 供按负载调校
DEFAULT_LOW_MFU = 0.15
MFU_KEY = "perf/actor_train_mfu"
MFU_PEAK_KEY = "perf/mfu_peak_tflops"
MFU_STEP_KEY = "rollout/step"
MFU_MIN_STEPS = 3 # 至少 3 个稳态 step 才下结论，避免早期波动误报


def mfu_summary(store: MetricStore) -> dict | None:
    series = store.metric_series([MFU_KEY, MFU_PEAK_KEY], x_key=MFU_STEP_KEY)
    steady = series[MFU_KEY]["y"][1:] # 排除 warmup 的 step 0
    if not steady:
        return None
    return dict(
        latest=steady[-1],
        mean=sum(steady) / len(steady),
        steps=len(steady),
        peak=series[MFU_PEAK_KEY]["y"][-1], # 分母随 summary 一起下发，前端可复算
    )


def _mfu_advisories(summary: dict | None, low_mfu: float) -> list[Advisory]:
    if low_mfu <= 0 or summary is None or summary["steps"] < MFU_MIN_STEPS:
        return []
    mean_mfu, peak = summary["mean"], summary["peak"]
    if mean_mfu >= low_mfu:
        return []
    return [
        Advisory(
            level="warning",
            message=(
                f"Model FLOPs utilization averaged {mean_mfu:.1%} of the device's {peak:g} TFLOP/
                s "
                f"over {summary['steps']} train steps — "
                "the training step is computing slowly, not waiting: this ratio counts actor train time
                only, "
                "so rollout stalls cannot depress it. Usual causes are activation recompute, a
                parallel split "
                "that leaves ranks idle, and small or ragged micro-batches"
            ),
        )
    ]


def compute_advisories(
    store: MetricStore,
    *,
    t0: float | None = None,
    t1: float | None = None,
```

```
    mfu: dict | None = None, # 允许 serve 层传入已算好的 summary，保证 tile 与规则不打架
    low_mfu: float = DEFAULT_LOW_MFU,
) -> list[Advisory]:
    out: list[Advisory] = _mfu_advisories(mfu if mfu is not None else mfu_summary(store), low_
    mfu)
    if not store.has_stream(Stream.ENGINE_SERIES):
        return out # 没有 sglang 数据时只返回 MFU 建议
    # 后续 concurrency / cache-hit / token-usage 建议逻辑保持不变（略）
    args = store.meta.args if store.meta else {}
    colocate = bool(args.get("colocate"))
    ...
```

评论区精华

1. H100 PCIe / NVL 峰值精度（yueming-yuan → 已解决）：reviewer 指出“This might not be accurate for all specs, e.g. H100 PCIe & H100 NVL”。作者在跟进 commit ca20caf 中新增 "H100 PCIE": 756.0 行，并把子串匹配升级为整词匹配（否则 torch 上报的连字符名 NVIDIA H100-PCIE-80GB 无法命中），同时论证 H100 NVL 是全尺寸 132-SM 芯片、不应降频。“先补数据、再修匹配机制”的处理顺序值得借鉴。
2. 0.15 阈值对大模型偏高（yueming-yuan → 已解决）：reviewer 给出实测：“I profiled long-context k2.5 training on GB300（MLA 架构、megatron 成熟实现），it can only reach ~17-18% mfu at best config”。作者回应“will add one choice for users, where they could set the boundary by themselves workload and expected MFU”，最终 commit b17ebc55 把常量改为 DEFAULT_LOW_MFU 并经 serve.py --low-mfu-threshold 暴露、传 0 关闭。这条讨论直接改变了产品形态：从硬编码规则变成可配置的工程决策。
3. Mixtral 风格专家宽度为 None 导致崩溃（yueming-yuan → 已解决）：reviewer 精准指出 moe_ffn_hidden_size 为 None 时 calculate_fwd_flops 内乘法会 TypeError。作者 commit 10b63396 在配置声明了专家数但缺少 moe_intermediate_size 时回退到 intermediate_size（dense 配置保持无专家宽度），把“崩溃”变成“正确数值”。
4. FYI 补充（yueming-yuan）：reviewer 明确标注“this one is not change request, just fyi”，仅提示阈值风险，作者随后主动给用户配置出口。

- H100 PCIe / NVL 峰值数据准确性 (correctness): 作者在 commit ca20caf 中新增 "H100 PCIE": 756.0 行，并把子串匹配升级为整词匹配（torch 上报的 "NVIDIA H100-PCIE-80GB" 才能命中），同时论证 H100 NVL 是全尺寸 132-SM 芯片不应降频，保留 989.0。
- LOW_MFU=0.15 阈值对大规模模型偏高 (design): 常量改为 DEFAULT_LOW_MFU，经 serve.py --low-mfu-threshold 暴露，传 0 关闭规则；PR body 亦明确标注该阈值为 placeholder 而非校准值，欢迎 reviewer 选择只发布指标与 tile。
- Mixtral 风格专家宽度为 None 导致崩溃 (correctness): 作者在 commit 10b63396 中当配置声明专家数但缺 moe_intermediate_size 时回退到 intermediate_size（dense 配置保持无专家宽度），把崩溃变成正确数值，并补充 test_mixtral_style_experts_sized_by_plain_intermediate_size 等测试。
- 阈值讨论的非变更请求说明 (question): 作者仍主动将阈值做成可配置 flag，保留默认 0.15 但允许按负载关闭或调校。

风险与影响

- 风险：
  1. 峰值表为人工维护的近似值：B300 沿用 B200 的 2250（基于芯片面积与 FP8/NVFP4 提升方向的推导），FP8/NVFP4 精度下实际可达算力更高，MFU 会系统性偏低；日期较新的芯片缺行时会静默不产出 MFU，需靠 --mfu-peak-tflops 手动补。
  2. FLOPs 模型只覆盖 base model 前向：--enable-mtp-training 的真实激活计算不进分子，MFU 被低估；该缺口同样影响既有 perf/actor_train_tflops，PR 中明确记录为已知缺口而非本次修复。
  3. flops_args_from_hf_config() 内 assert num_layers is not None：遇到无法识别的 HF 配置时会让 FSDP actor 直接抛异常——此前 FSDP 根本不计算 FLOPs、没有这条路径，属新引入的崩溃面。
  4. DEFAULT_LOW_MFU = 0.15 未校准：对 offload / recompute 重的 MoE 运行可能误报（实测健康 MoE 仅 5% 左右），已提供 --low-mfu-threshold 0 关闭规则作为缓解，但默认告警仍可能制造噪音。
  5. 与 #2027 的合并冲突：PR body 明确警告 #2027 重写同文件 compute_advisories，新规则是独立函数、合并时需并入其告警等级体系。
- 影响：
  1. 用户侧（训练工程师）：获得可直接判断“算得慢还是等数据”的 MFU 指标、看板 tile 与告警；FSDP 后端用户此前完全没有吞吐指标，本次补齐并统一到 megatron 口径。
  2. 系统侧：指标链路新增 perf/actor_train_mfu、perf/mfu_peak_tflops 两个 key，/api/advisory 响应结构新增 mfu 字段，前端随 views_timeline.js / app.js 更新；跨后端 FLOPs 口径一致性由 parity 测试锁定。
  3. 团队侧：为 dashboard 模块新增一个可配置告警规则和一份公共 statBox() 前端组件；后续阈值校准依赖真实运行数据积累，文档已沉淀公式与实测基线（8xH200：dense 25%/MoE 5%）。 - 风险标记：设备峰值表人工维护近似值，MFU 阈值未校准可能误报，MTP 等附加层 FLOPs 未计入分子，HF 配置断言可能引入新崩溃面，与 #2027 存在合并冲突

关联脉络

- PR #2027 （标题未知，见 PR body 冲突警告）：PR body 明确警告：#2027 重写同文件 compute_advisories，本 PR 新增的 MFU 规则是独立函数，合并时需以一行改动并入其告警等级体系，两者必须协调。
- PR #2386 fix: drop context parallelism from the FSDP backend: 同改 miles/backends/fsdp_utils/actor.py，FSDP 后端参数与执行路径的持续演进；本 PR 又为该后端接入了真实 FLOPs 计算与吞吐指标。
- PR #2024 dashboard: read open phase markers regardless of age: 同属 dashboard / observability 模块（miles/dashboard/store.py 数据链路），本 PR 的告警与指标视图继续沿用并扩展 store 的 metric_series 能力。