# PR #36699 完整报告

sgl-project/sglang

xpu: record per-model metrics to jsonl for nightly dashboard

合并时间：2026-09-01 10:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36699

## 执行摘要

- 一句话：XPU 夜间测试新增 per-model 指标 JSONL 记录与健康看板
- 推荐动作：值得精读。核心看点：① opt-in 零影响设计——env var 默认 None、全仓库无调用方，配合 append-only 与吞错，把"记录"与"决策"彻底分离，这是 CI 可观测性改造的安全姿势；② 双级记录（model 级 rich records + file 级 fallback）与 test_file 去重机制，兼顾数据丰富度与覆盖率；③ workflow 中直接内嵌 Python 渲染 dashboard 的做法，优点是单文件自包含，缺点是难测试。如果想复刻该模式，建议至少补一条 jsonl schema 的自动化测试，并统一 env var 的访问方式（全部走 Envs descriptor）。

## 功能与动机

PR body 明确指出现状痛点：xpu-ci-job-monitor.yml 目前只在 suite 级别（nightly-xpu-1-gpu / -2-gpu / -4-gpu）上报，而每个 suite 内部通过 run_suite.py 跑多个模型，monitor 只能看到 GitHub Actions 的聚合 job 状态——pass/fail，无法知道"哪个模型回归、回归多少、是否跌破精度门槛"。本 PR 是四 PR 系列的最小切片，目标是让监控降到 per-model 粒度，为后续 dashboard 提供 ref / actual / status / duration 数据底座。

## 实现拆解

1. 注册 opt-in 环境变量：python/sglang/srt/environ.py 的 Envs 类新增 SGLANG_TEST_METRICS_FILE = EnvStr(None)，挂在新 # CI reporting 分区下，遵循仓库 env-var conventions skill（所有 SGLANG_* 变量必须走 descriptor 而非裸 os.getenv）。默认 None 保证全仓库无调用方时零行为变化。
2. model 级指标写入：python/sglang/test/xpu/test_xpu_utils.py 的 write_results_to_github_step_summary 在写完 Markdown 表格后追加调用新函数 _append_metric_records；每个模型输出一行 JSON（kind=model），包含 accuracy、阈值、output_throughput、latency、num_prompts、num_threads、max_tokens、error 与 pass/fail 状态，test_file 取自 sys.argv[0] 供渲染端去重；同时 Step Summary 表头与每行新增 Prompts 列。
3. file 级兜底记录：python/sglang/test/ci/ci_utils.py 的 run_unittest_files 在 timings 输出之后、返回之前追加受保护的自动记录（kind=file）：按测试文件记录 pass/fail、duration 与 error，新增到 run_suite.py 的测试文件无需逐文件接线即可被 dashboard 覆盖。

4. workflow 上游接线：.github/workflows/nightly-test-intel.yml 为 1/2/4-GPU 三个 nightly 作业注入 SGLANG_TEST_METRICS_FILE=/sglang-checkout/nightly-xpu-{1,2,4}-gpu-metrics.jsonl，并新增 if: always() 的 actions/upload-artifact@v4 上传步骤（if-no-files-found: warn、retention 30 天），失败作业也能留下记录。
5. 消费端看板：.github/workflows/xpu-ci-job-monitor.yml 新增两个 job——nightly-xpu-status-dashboard 按 job 聚合最近 24 小时成功率并输出 shields.io badge（≥90% healthy、≥70% degraded、以下 critical 且 step 失败）；nightly-xpu-per-model-report 用 gh run list 解析最新 completed 的 nightly run，下载 metrics artifacts 后按 1x/2x/4x/8x 渲染 per-model 表格，8x 为未来预留。
6. 测试与验证配套：无新增自动化测试文件，PR 的 test plan 全为手动验证（py_compile、设置 / 未设置 env var 的 dry-run、grep 确认无既有调用方）；simple_eval_gsm8k_xpu_mixin.py 的 model_metrics 增加 num_prompts / num_threads / max_tokens 字段，为 dashboard 提供样本量上下文。

关键文件：

- python/sglang/test/xpu/test_xpu_utils.py（模块 XPU 测试工具；类别 test；类型 test-coverage；符号 _append_metric_records, write_results_to_github_step_summary）：数据管道核心：新增 _append_metric_records，把每个模型的精度 / 阈值 / 吞吐 / 延迟 / 状态写成一行 JSON 追加到 SGLANG_TEST_METRICS_FILE；同时 Step Summary 表格新增 Prompts 列。
- .github/workflows/xpu-ci-job-monitor.yml（模块 监控看板；类别 infra；类型 infrastructure；符号 badge, health, sort_key, load）：消费端：新增 nightly-xpu-status-dashboard（fleet 健康看板，90/70 阈值分级）与 nightly-xpu-per-model-report（下载 metrics artifact 并渲染 1x/2x/4x/8x per-model 表格），是整条数据管道的落点，也是本 PR 行数最大的变更。
- python/sglang/srt/environ.py（模块 环境变量；类别 source；类型 configuration）：按仓库 env-var conventions 注册 SGLANG_TEST_METRICS_FILE = EnvStr(None)，默认 None 是“未设置时零行为变化”承诺的根基；该文件是全局环境变量注册中心，后续所有平台复用此变量。
- python/sglang/test/ci/ci_utils.py（模块 测试编排；类别 test；类型 test-coverage；符号 run_unittest_files）：为未走 model 级写入器的测试文件提供 file 级兜底记录（kind=file），任何加入 run_suite.py 的测试自动被 dashboard 覆盖；该文件影响所有套件的测试编排链路。
- .github/workflows/nightly-test-intel.yml（模块 工作流配置；类别 infra；类型 infrastructure）：数据管道上游接线：为 1/2/4-GPU 三个 nightly 作业注入 SGLANG_TEST_METRICS_FILE 并新增 if: always() 的 artifact 上传步骤，保证失败作业也能留下指标。
- python/sglang/test/xpu/simple_eval_gsm8k_xpu_mixin.py（模块 基准评测；类别 test；类型 test-coverage；符号 test_gsm8k）：GSM8K mixin 的 model_metrics 补充 num_prompts / num_threads / max_tokens，让 dashboard 能结合样本量解读 accuracy 分数（200 个样本的 0.612 与 20 个样本的 0.612 统计意义不同）。

关键符号：_append_metric_records, write_results_to_github_step_summary, run_unittest_files, test_gsm8k

## 关键源码片段

### python/sglang/test/xpu/test_xpu_utils.py

数据管道核心：新增 _append_metric_records，把每个模型的精度 / 阈值 / 吞吐 / 延迟 / 状态写成一行 JSON 追加到 SGLANG_TEST_METRICS_FILE；同时 Step Summary 表格新增 Prompts 列。

```
def _append_metric_records(results: dict) -> None:
    """把每个模型的指标追加为一行 JSON 到 SGLANG_TEST_METRICS_FILE （若已设置）。

    该文件由 xpu-ci-job-monitor.yml 的 nightly XPU dashboard 消费，用于渲染
    每个模型的 ref / actual / status / duration 表格。所有写入失败都被静默吞掉，
    保证坏盘、权限错误或磁盘写满都不会让原本通过的测试变红。
    """
    path = envs.SGLANG_TEST_METRICS_FILE.get()
    if not path:
        # 环境变量未设置时直接返回，这是本 PR "默认零影响" 的基线。
        return

    # sys.argv[0] 是测试脚本路径（例如 `python3 test_foo.py` 直接运行时）；
    # 渲染端按 test_file 分组聚合，file 级兜底记录不会与 model 级记录重复计数。
    test_file = os.path.basename(sys.argv[0]) if sys.argv and sys.argv[0] else ""

    try:
        # 以追加模式打开，绝不截断或覆盖，同一 runner 上并发进程不会互相踩踏。
        with open(path, "a") as f:
            for model, metrics in results.items():
                record = {
                    "kind": "model", # 与 ci_utils.py 里的 file 级记录区分用途
                    "test_file": test_file,
                    "model": model,
                    "accuracy": metrics.get("accuracy"),
                    "accuracy_threshold": metrics.get("accuracy_threshold"),
                    "output_throughput": metrics.get("output_throughput"),
                    "output_throughput_threshold": metrics.get(
                        "output_throughput_threshold"
                    ),
                    "latency": metrics.get("latency"),
                    "num_prompts": metrics.get("num_prompts"),
                    "num_threads": metrics.get("num_threads"),
                    "max_tokens": metrics.get("max_tokens"),
                    "error": metrics.get("error", ""),
                    "status": "pass" if not metrics.get("error") else "fail",
                }
                f.write(json.dumps(record) + "\n")
```

```
    except OSError:
        # CI 安全底线：记录失败绝不能影响测试结果。
        pass
```

### python/sglang/test/ci/ci_utils.py

为未走 model 级写入器的测试文件提供 file 级兜底记录（kind=file），任何加入 run_suite.py 的测试自动被 dashboard 覆盖；该文件影响所有套件的测试编排链路。

```
# 完全受保护的自动记录：`SGLANG_TEST_METRICS_FILE` 默认未设置，对所有非 XPU
# nightly 套件都是零差异；`OSError` 同样被吞掉，坏文件系统不能把通过的运行变红。
# 新增到 run_suite.py 的测试文件无需逐个接线，这里会自动拾取。
metrics_path = os.environ.get("SGLANG_TEST_METRICS_FILE")
if metrics_path:
    passed_set = set(passed_tests)
    failed_reasons = dict(failed_tests)
    try:
        with open(metrics_path, "a") as f:
            for fname, elapsed in file_elapsed.items():
                record = {
                    "kind": "file", # 文件级兜底，渲染端按 test_file 去重
                    "test_file": os.path.basename(fname),
                    "status": "pass" if fname in passed_set else "fail",
                    "duration": round(elapsed, 2),
                }
                if fname in failed_reasons:
                    record["error"] = failed_reasons[fname]
                f.write(json.dumps(record) + "\n")
    except OSError:
        pass
```

## 评论区精华

本 PR 没有公开 review 评论，只有一条 issue 评论：

> "@MingxuZh could you please help review this one?" —— mingfeima

最终由 mingfeima 本人 APPROVED 并合入。真正的设计讨论藏在 PR body 与提交历史中：

- PR body 的 CI safety 三原则构成了整条管道的设计基线：① 未设置 env var 时无行为变化；② 所有 OSError 被吞掉，写入失败不能把通过的测试变红；③ 追加模式写入，绝不截断覆盖，并发 worker 互不踩踏。其中"无行为变化"的承诺只对 jsonl 写入严格成立——Markdown Step Summary 表格实际上新增了 Prompts 列，属于承诺之外的静态变化。
- commit 2a62b54 明确记录 schema 分歧是有意为之：num_prompts / num_threads / max_tokens 先只在 XPU 侧落地，test_ascend_utils.py 的对应改动"left for the Ascend maintainer"，避免一次性跨平台大改。
- commit 45c6c38 引入 file 级自动记录时为 model 级记录打 test_file 戳，代码注释说明渲染端按文件分组、避免 file 级 fallback 行与 model 级 rich 记录重复计数——这是双写路径的去重设计。

- PR body 声称"consumer 是 follow-up PR、本 PR 只是 plumbing"，但提交 77751a01 实际把 dashboard 也合入了（范围在演进中扩大），body 的描述已与合入内容不符。
- 暂无高价值评论线程

## 风险与影响

- 风险：
  1. "未设置时零变化"承诺不完整：test_xpu_utils.py 的 Markdown 表头与每行都新增了 Prompts 列，即使 SGLANG_TEST_METRICS_FILE 未设置，$GITHUB_STEP_SUMMARY 的表格内容也始终变化；任何依赖旧列序解析 Step Summary 的脚本（包括 Ascend 侧镜像代码）可能受影响。
  2. env var 访问方式不一致：test_xpu_utils.py 走 envs.SGLANG_TEST_METRICS_FILE.get() descriptor，而 ci_utils.py 用 os.environ.get("SGLANG_TEST_METRICS_FILE")，违反 PR 自己宣称的 env-var conventions；未来若对变量做验证、别名或废弃处理，两条路径行为会分裂。
  3. 双写路径依赖渲染端去重：model 级记录与 file 级记录同时写入同一 jsonl，dashboard 若未严格按 test_file 分组会出现重复计数；test_file 取自 sys.argv[0]，在 pytest 或 runner import 场景下可能是空串或非预期值。
  4. 错误完全静默：except OSError: pass 意味着磁盘满、权限错误时 dashboard 数据静默缺失，故障不可见。
  5. 大段内嵌 Python 无测试守护：xpu-ci-job-monitor.yml 新增 363 行 heredoc 内嵌脚本（badge/health/sort_key 等），无单元测试、无 lint，shields.io badge 还依赖外网可达性；nightly-xpu-per-model-report 依赖 gh run list 解析与 artifact 下载，任一环节失败只靠 continue-on-error 兜底。
  6. 缺少自动化测试：PR 的验证全部为手动清单，jsonl schema、Step Summary 格式、双路径去重均无 CI 用例守护，后续改动容易静默破坏管道。 - 影响：用户与产品运行时零影响：SGLANG_TEST_METRICS_FILE 默认 None，仓库内无既有调用方，environ.py 仅新增一个配置项注册。对系统的影响集中在 CI 侧：XPU nightly 三个作业（1/2/4-GPU）每次运行会额外写一个 jsonl 并上传 artifact（保留 30 天），xpu-ci-job-monitor.yml 新增两个 ubuntu-latest job 的渲染开销。对团队的价值是监控粒度从套件级降到模型级：能精确看到哪个模型、以多少吞吐 / 精度、对哪个门槛失败，并首次为 XPU nightly 提供历史可追溯的指标文件。该管道也为后续平台（Ascend、未来 8-GPU 套件）提供了可复制的模式；ci_utils.py 的 file 级记录对所有套件生效，其他 nightly 只要设置同一 env var 即可复用。 - 风险标记：未设置 env var 时表格结构仍变化，缺少自动化测试覆盖，双写路径依赖渲染端去重，CI 脚本内嵌大段 Python 无测试守护，env var 访问方式不一致

## 关联脉络

- PR #35500 [CI/NPU] Isolate multi-node tests by run_id to prevent concurrent-run…：同为硬件（NPU/XPU）nightly CI 健壮性演进：按 run_id 隔离并发运行与本 PR 的 append-only 并发写入防护目标一致，都是让 nightly 结果可信的基础设施工作。

- PR #34074 [CI] Move tests onto the right CI stages: CI 测试分级与 workflow 矩阵化，改动集中在同一测试编排链路（run_suite.py / ci_utils.py 相关）；本 PR 在 ci_utils.py 的 run_unittest_files 中新增 file 级记录，二者在共同演进测试执行框架。