PR #35030 完整报告

sgl-project/sglang

Add bit-exact guard for extra_buffer_lazy

合并时间：2026-08-16 23:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35030

执行摘要

- 一句话：为 extra_buffer_lazy 新增 bit-exact 缓存守卫
- 推荐动作：值得精读，尤其是关注测试方法论而非实现细节：为什么 extra_buffer_lazy 需要独立测试类而不是在现有类上加一个 flag；revert-then-red 如何把「守卫确实能检测缺陷」变成可验证的硬证据；KL_DIV_THRESHOLD、TRACK_INTERVAL、PAGE_SIZE、MAX_NEW_TOKENS 等常量如何与模型特性（SWA 窗口、page 边界、chunked prefill）相互绑定。对于想为 speculative decoding、MTP、DSpark 等路线图项补守卫的开发者，这是一个可直接复制的模板。

功能与动机

PR body 明确指出 extra_buffer_lazy 此前完全没有 bit-exact 覆盖，且它是「a different code path rather than the same code under a flag」：请求只持有 1 个 ping-pong 槽位而不是 2 个，第二个槽位仅在跨 track 边界的 forward 上分配并释放，checkpoint 因此落在逐 forward 选择的槽位上，选错就会恢复另一个请求的 conv 窗口。关联 issue #34899 确立了「任何非零 KL 都是 KV/ 状态复用 bug」的契约，并把 mamba extra_buffer_lazy 列为路线图待覆盖项，因此需要为这条独立路径单独建立回归守卫。

实现拆解

1. 参数化启动参数：_base_args() 增加 mamba_strategy: str = "extra_buffer" 参数，--mamba-radix-cache-strategy 从硬编码字面量改为变量。这让新测试类只需换一个策略名即可复用整套启动参数（--attention-backend fa4、--page-size 128、--mamba-track-interval 128、--enable-deterministic-inference 等），避免复制粘贴带来的配置漂移。
2. 新增 lazy 测试类：TestUnifiedHybridLazyBitExact(TestUnifiedHybridBitExact) 只重写 setUpClass，以 _base_args("extra_buffer_lazy") + ["--chunked-prefill-size", "16384"] 启动服务，其余全部继承基类三个测试方法（test_logprobs_match、test_prefill_cache_hit、test_decode_cache_hit）。--chunked-prefill-size 16384 的作用是保证 decode 过程中跨过 track 边界，让 lazy 槽位的分配与释放路径真正被走到。
3. 文档与 CI 预算：文件头部 KL 对照表新增 lazy 行（SM100 上 revert #34184 时为 5.56e-06，修复后 0.0），并说明该值与同会话非 lazy 路径的 1.18e-05 应解读为「both fire」；register_cuda_ci(est_time=600 → 790) 反映新增用例的实际耗时。

配套情况：这是纯测试与 CI 配套变更，无配置、schema 或部署改动。测试注册在 1-gpu-large（SM90）的 base-b stage，PR 标签带 run-ci 与 run-ci-extra。

关键文件：

- test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_hybrid_bitexact.py（模块 缓存测试；类别 test；类型 test-coverage；符号 _base_args, TestUnifiedHybridLazyBitExact, setUpClass）：唯一的变更文件：新增 TestUnifiedHybridLazyBitExact 类为 extra_buffer_lazy 策略提供 bit-exact KL 守卫，同时把 _base_args 参数化并更新 CI 耗时估计，是本次 PR 的全部价值所在。

关键符号：_base_args, TestUnifiedHybridLazyBitExact.setUpClass

关键源码片段

test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_hybrid_bitexact.py

唯一的变更文件：新增 TestUnifiedHybridLazyBitExact 类为 extra_buffer_lazy 策略提供 bit-exact KL 守卫，同时把 _base_args 参数化并更新 CI 耗时估计，是本次 PR 的全部价值所在。

```
def _base_args(mamba_strategy: str = "extra_buffer") -> list[str]:
    # 参数化 radix cache 策略，让不同测试类复用同一套启动参数，
    # 只需要切换 --mamba-radix-cache-strategy 这一个配置键。
    return [
        "--trust-remote-code",
        "--attention-backend", "fa4",
        "--page-size", str(PAGE_SIZE),
        "--mamba-radix-cache-strategy", mamba_strategy,
        "--swa-full-tokens-ratio", "0.1",
        "--mamba-full-memory-ratio", "0.1",
        # 0.85 是从 4-GPU B200 测试延续下来的值，在 80 GB 显卡上会 OOM：
        # 静态池只留约 19 GB 给 prefill graph、fa4 workspace 和 chunked
        # prefill 激活，0.6 才是这个配置需要的值。
        "--mem-fraction-static", "0.6",
        "--mamba-track-interval", str(TRACK_INTERVAL),
        "--enable-deterministic-inference",
    ]


class TestUnifiedHybridLazyBitExact(TestUnifiedHybridBitExact):
    """Same exactness bar on the lazy extra-buffer strategy.

    Lazy 是一段独立代码路径，而不是同一个代码换了个 flag：请求只持有
    1 个 ping-pong 槽位而不是 2 个，第二个槽位只在跨 track 边界的 forward
    上临时分配、用完即释放。因此 checkpoint 落在每次 forward 临时选择的
    槽位上，而不是请求整个生命周期拥有的槽位，选错就会恢复另一个请求的
    conv 窗口。

    验收方式和它的基类一致（revert-then-red）：revert #34184 后，本类的
```

```
    test_prefill_cache_hit 会读到 5.56e-06，远高于 1e-9 门槛，所以它守卫的
    是 lazy 路径上同一类状态复用 bug。
    """

    @classmethod
    def setUpClass(cls):
        cls.model = _MODEL_PATH
        cls.base_url = DEFAULT_URL_FOR_TEST
        other_args = _base_args("extra_buffer_lazy") + [
            "--chunked-prefill-size", "16384",
        ]
        # 加大 chunked prefill，让 decode 跨过 track 边界，
        # 从而触发 lazy 槽位的分配与释放路径。
        if _MODEL_REVISION:
            other_args += ["--revision", _MODEL_REVISION]
        cls.process = popen_launch_server(
            cls.model,
            cls.base_url,
            timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
            other_args=other_args,
            env={**os.environ, "SGLANG_ENABLE_UNIFIED_RADIX_TREE": "1"},
        )
```

评论区精华

该 PR 没有任何 review 评论（review_comments_count = 0）。PR body 提供了最关键的论证：lazy 路径与常规 extra_buffer 是不同代码路径，不能复用同一测试类加 flag，必须单独起服务实例；灵敏度用 revert-then-red 证明——revert #34184 后 lazy 类读到 5.56e-06，非 lazy 类同会话读到 1.18e-05，因此「两个都触发」，不能说 lazy 守卫更弱；实测耗时 655s（新类仅 129s，因为复用了 warm model），据此把 est_time 从 600 调到 790。issue 评论区只有作者发的 /rerun-test 指令与 GitHub Actions 机器人报告的通过结果，没有技术分歧或未解决问题。

- 暂无高价值评论线程

风险与影响

- 风险：风险集中在测试有效性而非生产代码：1）CI 时长与预算——base-b stage 的 est_time 增加约 32%（600 → 790s），实测全文件 655s、新类只占 129s，预算余量约 20%，若同 stage 其他用例继续变慢仍有超时风险；2）架构盲区——lazy 行的 revert 实测只在 SM100（B200）上完成，SM90（CI 实际硬件）标记为 n/a，若两台架构上 lazy 槽位行为不同，该守卫在 CI 上可能「测不到」它声称覆盖的 bug，属于已记录但未验证的盲区；3）模型确定性依赖——守卫前提是 Inkling@test 在 --enable-deterministic-inference 下 prefill 与 decode 逐位一致，checkpoint 或 revision 更新可能破坏该前提并产生假阳性。文件层面风险点集中在 test_unified_radix_cache_kl_hybrid_bitexact.py 内新增类的启动参数组合。

- 影响：对用户无直接影响：没有任何生产代码或部署配置变更。对系统的影响是每个提交在 1-gpu-large（SM90）的 base-b stage 新增一个约 129s 的 GPU 用例，CI 预算从 600s 上调至 790s。对团队而言，extra_buffer_lazy 这条容易在槽位复用上出错的策略首次获得逐位一致性护栏，防止 #34184 这类状态复用 bug 在 lazy 路径上复发，并补齐 issue #34899 路线图中的一块覆盖空白。
- 风险标记：CI 时长预算上调 32%, SM90 上 lazy 信号未验证，依赖 Inkling@test 确定性

关联脉络

- PR #34763 [Spec] Support mamba-radix-cache-strategy extra_buffer_lazy with DFLASH: 同属 mamba extra_buffer_lazy 策略的功能线；该 PR 为 DFLASH 接入此策略，本 PR 为同一策略补上 bit-exact 正确性守卫。