PR #36219 完整报告

sgl-project/sglang

[Performance] Tune FlashInfer EXTEND for DP prefill

合并时间：2026-08-25 23:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36219

执行摘要

- 一句话：EXTEND 调优按实际调度器缓冲预热，DP prefill 吞吐提升约 6%
- 推荐动作：建议精读 flashinfer_autotune.py 中 token 预算来源与 base_runner.py 断言放宽两处改动，重点学习「预热形状必须对应生产负载的真实 per-rank 缓冲」这一性能分析思路。如果部署并启用 SGLANG_FLASHINFER_AUTOTUNE_EXTEND，建议先在小流量验证多模态与投机组合，再逐步扩大。

功能与动机

在 DP attention prefill worker 中，显式 FlashInfer EXTEND 自动调优预热原先使用 max_prefill_tokens，而这是全局 DP token 预算而不是单 rank 的实际负载。当全局 / 局部 chunk size 为 32768/8192 时，每个 rank 预热的是 32768-token 形状，而非生产的 8192-token 调度器缓冲，最终导致生产 FlashInfer MoE EXTEND 桶未被调优，默认 heuristic 明显更慢。该路径还跳过了投机 PD prefill 目标与多模态生成包装器的纯文本服务场景。

实现拆解

实现按三步推进：

1. 调整 EXTEND dummy 的 token 预算（flashinfer_autotune.py）：把 num_tokens 从 mr.server_args.max_prefill_tokens 改为 mr.server_args.max_prefill_buffer_tokens() or mr.server_args.max_prefill_tokens。max_prefill_buffer_tokens() 返回单 rank 调度器缓冲上限，在 chunked prefill 禁用时返回 0，因此用旧上限兜底。新增 is_pd_prefill_target 判定（get_disagg().disaggregation_mode == "prefill" and not mr.is_draft_worker），使投机 runner 的跳过条件只对“非 PD prefill 目标”生效；同时删除 mr.model_config.is_multimodal 的跳过分支，让多模态生成包装器也能执行纯文本 EXTEND dummy。
2. 放宽 _dummy_run 的 EXTEND 断言（base_runner.py）：提前计算 _is_pd_prefill_target，把 extend_num_tokens_per_req 的断言从“必须非投机”调整为“普通或 PD prefill 目标”，从而允许 PD prefill 目标使用 packed EXTEND 形状；普通投机目标仍被拒绝，保持原有契约。
3. 补充 CPU 回归测试（test_flashinfer_autotune.py）：test_packed_speculative_extend_is_limited_to_pd_prefill_target 用参数化验证带 spec

的 EXTEND dummy 只在 PD prefill 目标可接受；test_chunked_prefill_disabled_uses_legacy_token_ceiling 验证 max_prefill_buffer_tokens() 为 0 时回退到 max_prefill_tokens（32768），并断言 _alloc_dummy_decode_buffers 收到正确形状。测试注册到 base-a-test-cpu CI 套件。

关键文件：

- python/sglang/srt/model_executor/runner/flashinfer_autotune.py（模块 自动调优；类别 source；类型 core-logic；符号 maybe_flashinfer_autotune_extend）：核心改动：EXTEND 自动调优的 token 预算从全局 max_prefill_tokens 改为 per-rank max_prefill_buffer_tokens()，并为 PD prefill 投机目标与多模态纯文本预热放开跳过限制。
- python/sglang/srt/model_executor/runner/base_runner.py（模块 执行器；类别 source；类型 data-contract；符号 BaseRunner._dummy_run）：调整 _dummy_run 对 EXTEND dummy 的断言，允许 PD prefill 投机目标使用 packed EXTEND 形状，普通投机目标保持拒绝。
- test/registered/unit/model_executor/runner/test_flashinfer_autotune.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_packed_speculative_extend_is_limited_to_pd_prefill_target, test_chunked_prefill_disabled_uses_legacy_token_ceiling）：新增两个 CPU 回归测试，锁定 review 阶段确认的边界行为：packed 投机 EXTEND 仅限 PD prefill 目标、chunked prefill 禁用时回退旧 token 上限。

关键符号：maybe_flashinfer_autotune_extend, BaseRunner._dummy_run, test_packed_speculative_extend_is_limited_to_pd_prefill_target, test_chunked_prefill_disabled_uses_legacy_token_ceiling

关键源码片段

python/sglang/srt/model_executor/runner/flashinfer_autotune.py

核心改动：EXTEND 自动调优的 token 预算从全局 max_prefill_tokens 改为 per-rank max_prefill_buffer_tokens()，并为 PD prefill 投机目标与多模态纯文本预热放开跳过限制。

```
def maybe_flashinfer_autotune_extend(runner, *, decode_num_tokens):
    """EXTEND 形状 dummy 的 FlashInfer 自动调优入口（由功能开关控制）。"""

    if not envs.SGLANG_FLASHINFER_AUTOTUNE_EXTEND.get():
        return

    mr = runner.model_runner
    # 优先使用 per-rank 调度器缓冲，这才是生产负载的真实 token 数；
    # 禁用 chunked prefill 时 max_prefill_buffer_tokens() 返回 0，回退旧上限。
    num_tokens = (
        mr.server_args.max_prefill_buffer_tokens() or mr.server_args.max_prefill_tokens
    )
    if num_tokens <= (decode_num_tokens or 0):
        return # decode 形状的调优已覆盖这些小桶

    # PD prefill 目标 worker 没有 draft 状态，可以安全保留 EXTEND 模式；
```

```
    # 普通投机 runner 会被 _dummy_run 强制为 TARGET_VERIFY，这里仍跳过。
    is_pd_prefill_target = (
        get_disagg().disaggregation_mode == "prefill" and not mr.is_draft_worker
    )
    if not mr.is_generation or (
        mr.spec_algorithm.is_speculative() and not is_pd_prefill_target
    ):
        return

    # 多模态生成包装器也可以跑纯文本 dummy；若模型不兼容，会由显式 opt-in 暴露错误。
    if mr.attn_backend.extend_dummy_seqs_capped_by_req_pool:
        pool_size = mr.req_to_token_pool.size
        num_tokens_per_req = (num_tokens + pool_size - 1) // pool_size
    else:
        # 打包的 dummy 可能调出更差的 tactic，只有后端会崩溃时才打包；
        # None 让后端在 _dummy_run 里使用自己的 seq_len_fill_value。
        num_tokens_per_req = None
    per_req = num_tokens_per_req or 1
    batch_size = (num_tokens + per_req - 1) // per_req
    num_tokens = batch_size * per_req

    buffers = runner._alloc_dummy_decode_buffers(
        batch_size,
        num_tokens_per_req=per_req,
        allocate_logits_buffer=False,
    )
    # 拿到正确形状的静态缓冲区后，再构造一次 EXTEND forward 交给自动调优执行。
```

python/sglang/srt/model_executor/runner/base_runner.py

调整 _dummy_run 对 EXTEND dummy 的断言，允许 PD prefill 投机目标使用 packed EXTEND 形状，普通投机目标保持拒绝。

```
    num_tokens_per_req = 1
    # PD prefill 目标 worker 的 token pool 没有 SpeculativeState，TARGET_VERIFY
    # dummy 会触发线性注意力后端的池类型断言，因此这里允许它保留 EXTEND 模式。
    _is_pd_prefill_target = (
        get_disagg().disaggregation_mode == "prefill" and not mr.is_draft_worker
    )
    if mr.spec_algorithm.is_speculative() and not _is_pd_prefill_target:
        if mr.is_draft_worker:
            assert mr.spec_algorithm.supports_target_verify_for_draft(), (
                "This should not happen"
            )
        capture_forward_mode = ForwardMode.TARGET_VERIFY
        num_tokens_per_req = mr.decode_num_tokens_per_req()
    if extend_num_tokens_per_req is not None:
        assert capture_forward_mode == ForwardMode.EXTEND and (
            not mr.spec_algorithm.is_speculative() or _is_pd_prefill_target
        ), "extend_num_tokens_per_req requires an ordinary or PD-prefill target EXTEND dummy"
```

```
        num_tokens_per_req = extend_num_tokens_per_req
    num_tokens = batch_size * num_tokens_per_req
```

test/registered/unit/model_executor/runner/test_flashinfer_autotune.py

新增两个 CPU 回归测试，锁定 review 阶段确认的边界行为：packed 投机 EXTEND 仅限 PD prefill 目标、chunked prefill 禁用时回退旧 token 上限。

```
def test_chunked_prefill_disabled_uses_legacy_token_ceiling():
    model_runner = SimpleNamespace(
        server_args=SimpleNamespace(
            max_prefill_buffer_tokens=Mock(return_value=0),
            max_prefill_tokens=32768,
        ),
        is_generation=True,
        is_draft_worker=False,
        spec_algorithm=SimpleNamespace(is_speculative=lambda: False),
        attn_backend=SimpleNamespace(extend_dummy_seqs_capped_by_req_pool=False),
        canary_manager=None,
    )
    runner = SimpleNamespace(
        model_runner=model_runner,
        _alloc_dummy_decode_buffers=Mock(return_value=object()),
        _dummy_run=Mock(),
    )
    # chunked prefill 关闭时 max_prefill_buffer_tokens() 返回 0，
    # 应回退到旧上限 32768，而不是静默跳过显式 opt-in。
    with (
        patch.object(
            flashinfer_autotune.envs.SGLANG_FLASHINFER_AUTOTUNE_EXTEND,
            "get",
            return_value=True,
        ),
        patch.object(
            flashinfer_autotune,
            "get_disagg",
            return_value=SimpleNamespace(disaggregation_mode="prefill"),
        ),
        patch.object(flashinfer_autotune, "run_flashinfer_autotune_forward"),
        patch.object(flashinfer_autotune.torch.cuda, "empty_cache"),
    ):
        flashinfer_autotune.maybe_flashinfer_autotune_extend(
            runner, decode_num_tokens=128
        )

    runner._alloc_dummy_decode_buffers.assert_called_once_with(
        32768,
        num_tokens_per_req=1,
        allocate_logits_buffer=False,
    )
```

评论区精华

该 PR 的 review 数据中没有可用的评论文本（review_comments_count=0），只有作者自己的 /tag-and-rerun-ci 触发 CI。PR body 透露了 review 阶段的两个发现并已通过测试锁定："packed speculative EXTEND is accepted only for a PD prefill target"与"disabling chunked prefill falls back to max_prefill_tokens instead of silently skipping the explicit autotune opt-in"。可见讨论焦点集中在「哪些投机配置允许 EXTEND dummy」和「chunked prefill 关闭时的回退语义」，这两点最终通过断言修改和两个 CPU 回归测试闭环。

- EXTEND 自动调优的 token 预算与投机目标覆盖 (design): 以 per-rank max_prefill_buffer_tokens() 为预算，chunked prefill 关闭时回退旧上限；PD prefill 目标允许 EXTEND，普通投机目标仍拒绝。

风险与影响

- 风险：主要风险有四：一是 BaseRunner._dummy_run 是 CUDA graph 捕获和多种预热路径的公共入口，放宽断言后需关注其他投机 /PP 组合是否会被意外放行；二是多模态行为从"静默跳过"变为"显式失败"，若某些多模态模型不支持纯文本 dummy，开启 SGLANG_FLASHINFER_AUTOTUNE_EXTEND 的用户会在启动阶段遇到失败；三是性能结论只在 GB300 + Qwen3.5-397B-A17B-NVFP4 + FlashInfer A2A + CuteDSL MoE 单配置下验证，其他 GPU/ 模型 / 后端不一定能复现 5%-7% 收益；四是新增测试是 CPU 模拟，不执行真实 FlashInfer 内核，无法覆盖 CUDA 路径上的形状 / 内存问题。
- 影响：对开启了 SGLANG_FLASHINFER_AUTOTUNE_EXTEND 且使用 FlashInfer 的 DP prefill（含 PD 投机）部署，EXTEND 调优桶将与真实负载匹配，实测 TPS/chip 提升约 5%-7%（C4-C64 区间），C128 因候选波 CV 超 1% 门槛被排除在性能声明外；GSM8K 准确率 0.985、平均接受长度 4.4966、CUDA graph 请求分数 1.0，无回归。对未开启该环境变量的用户无行为变化；对禁用 chunked prefill 的用户沿用旧上限，避免形状回退。仓库新增约 94 行 CPU 单元测试并挂入 CI。
- 风险标记：核心 runner 路径变更，多模态预热行为从跳过改为显式失败，性能结论限于单一配置，新增测试为 CPU 模拟

关联脉络

- PR #36231 [DeepGEMM] Deduplicate JIT precompile across local ranks: 同为启动 / 预热阶段的优化：跨 rank 去重 DeepGEMM JIT 预编译，与本 PR 的 EXTEND autotune 预热同属降低重复预热成本、对齐生产负载的方向。
- PR #35672 [AMD] Enable draft_extend CUDA graph for HIP DSA backend: 同样触及投机场景下的 EXTEND 路径（draft_extend），本 PR 为 PD prefill 投机目标放宽了 EXTEND dummy 的限制，两者在投机 EXTEND 的契约上相互关联。