

PR #32888 完整报告

sgl-project/sglang

[AMD][gfx95] Fill the chunked-prefill compute budget exactly

合并时间: 2026-09-14 15:35

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/32888>

执行摘要

- 一句话: gfx95 上把 prefill 计算预算按原始 token 计费, batch 精确填满 chunked-prefill-size
- 推荐动作: 值得精读, 但重点不在 diff 的行数。建议按以下顺序看:
 1. `PrefillAdder._update_prefill_budget` 新增的 `compute_charge` 参数——这是全 PR 的设计核心, 理解“KV 预算按 page、计算预算按 token”的分工边界, 以及为什么两个计算预算必须同时切换 (否则先归零者会留下取整余量)。
 2. `add_one_req` 的新 `elif self.exact_chunk_fill`: 分支——注意 `chunk_tokens_limit` 而非 `rem_chunk_tokens` 的选择、`truncation_align_size` 对齐、以及 `_check_prefill_tile_budget` 的插入位置, 这三点决定了它和 SWA/tilebudget 特性的边界。
 3. 最后一个 merge commit 对上游 `raw_extend_input_len` 的处理, 是“让本地改动贴合上游口径”的良好实践样本。

改进建议: 为 `exact_chunk_fill` 补一个可直接断言的单元测试 (例如 `page_size=64`、`chunk=16384`、prompt 混合, 断言 `rem_chunk_tokens` 归零且 extend 总长恰为 16384), 因为当前等价性与正确性完全依赖作者的一次性场景重放, 无法防止后续回归。

功能与动机

PR body 指出两个独立的 page 取整在持续泄漏 chunk 预算: `_update_prefill_budget` 把页取整后的 extend 长度记到 `rem_chunk_tokens / rem_input_tokens`, `add_one_req` 又把最后一段截断向下 floor 回 page, 于是 `page_size=64` 时 16384 的 chunk 只跑到 16328 (trace 里 toks=16350)。作者强调这个缺口在 gfx95 上代价很高: aiter MLA absorb bmm 依据编译期 `constexpr M % BLOCK_SIZE_M` 选择 EVEN_MN 特化, M 不对齐等于换了一个编译产物而不是同一个 kernel 跑残块, 实测 dense GEMM 从 163.5 ms 涨到 136.6 ms (反向即收益), 所以要在调度侧把 M 对齐这件事做对。

实现拆解

1. 新增开关与硬件门控: `python/sglang/srt/envron.py` 在“Scheduler token budgeting and admission”段落新增 `SGLANG_EXACT_CHUNK_FILL = EnvBool(True)`; `schedule_policy.py` 新增模块级 `@lru_cache(maxsize=1) def _use_exact_chunk_fill()`, 组合环境开关与从 `sglang.srt.utils` 新导入的 `is_gfx95_supported()`, 并在 `PrefillAdder.__init__` 落成实例字段 `self.exact_chunk_fill = _use_exact_chunk_fill()` and `dllm_config is None`, 显式把 dLLM 路

径排除在外。门控放实例级而非调用级，是因为后续 4 处调用点都要读它，缓存后零开销。

2. 计算预算与 KV 预算解耦：PrefillAdder._update_prefill_budget 新增关键字参数 compute_charge: Optional[int] = None，缺省回落为 ceil_paged_tokens(extend_input_len)。rem_total_token_offset / cur_rem_token_offset / rem_swa_token_offset 仍用取整长度（KV pool 分配粒度就是 page，少算会 OOM），而 rem_input_tokens、rem_dllm_tokens、rem_chunk_tokens 三个计算预算改用 compute_charge。注释特意说明两个计算预算必须同时切换：它们通常配成同一数值，只改一个会让另一个先归零，提前掐断 admission，把取整余量留成废预算。

3. 整条装入判定与计费：add_one_req 把原来的 input_tokens 拆成 raw_input_tokens 与 ceil_paged_tokens(raw_input_tokens)，新增 chunk_fit_tokens = raw_input_tokens if self.exact_chunk_fill else input_tokens 用于“能否整条装入”的比较，避免一条本可整装下的 prompt 仅因 ceil 后多出几十 token 就被硬拆成第二段；该路径传 compute_charge=raw_input_tokens。add_chunked_req 与 add_req_state 同步传 compute_charge=req.extend_range.length / trunc_len，保证续块和多段截断也不带取整损耗。

4. 新增 exact-fill 截断分支：当 chunk_fit_tokens > chunk_tokens_limit 且开启 exact-fill 时，新分支直接 trunc_len = chunk_tokens_limit（注释说明用 limit 而非 rem_chunk_tokens，是为了让 SWA 限流后的 chunk 上限继续生效），再按 truncation_align_size 向下对齐、走 _check_prefill_tile_budget(trunc_len)，然后 req.set_extend_range、can_run_list.append、new_chunked_req、_req_inc_lock_ref、_update_prefill_budget(..., compute_charge=trunc_len)。这一步把原来“floor 回 page”的截断换成精确填满，是 M 对齐的最后一环。

5. 测试与验证配套：本次仅改 2 个源码文件，没有新增或修改任何测试（changed_files_count=2，无 test 目录变更）。验证依赖两条：PR body 中 48 组 admission 场景（page_size 1/8/64 × chunk 2048/16384 × 四种 prompt 组合，114 次 admission）在关开关时与 upstream/main 预算计数器逐字节一致、强制开启时只移动 73 行、page_size=1 两种情况完全相同；以及 MI35x 上的 CI 作业（amd-bot 明确点名该路径在 stage-b-test-1-gpu-small-amd-mi35x 等 3 个 job 上被实际执行并通过）。

关键文件：

- python/sglang/srt/managers/schedule_policy.py（模块 调度器；类别 source；类型 core-logic；符号 _use_exact_chunk_fill, PrefillAdder.init, PrefillAdder._update_prefill_budget, PrefillAdder.add_one_req）：全部核心逻辑所在：新增 _use_exact_chunk_fill() 门控、PrefillAdder.exact_chunk_fill 实例开关、_update_prefill_budget 的 compute_charge 双口径计费，以及 add_one_req 的 exact-fill 截断分支；同时影响所有后端共享的 admission 行为。
- python/sglang/srt/envron.py（模块 运行配置；类别 source；类型 configuration；符号 SGLANG_EXACT_CHUNK_FILL）：新增全局开关 SGLANG_EXACT_CHUNK_FILL = EnvBool(True)，是 gfx95 上默认开启、其他平台可通过环境变量整体回退的唯一控制点；注释直接指向 PrefillAdder.exact_chunk_fill，是理解门控语义的入口。

关键符号: `_use_exact_chunk_fill`, `PrefillAdder.init`, `PrefillAdder._update_prefill_budget`, `PrefillAdder.add_one_req`, `PrefillAdder.add_chunked_req`, `PrefillAdder.add_req_state`

关键源码片段

python/sglang/srt/managers/schedule_policy.py

全部核心逻辑所在: 新增 `_use_exact_chunk_fill()` 门控、`PrefillAdder.exact_chunk_fill` 实例开关、`_update_prefill_budget` 的 `compute_charge` 双口径计费, 以及 `add_one_req` 的 `exact-fill` 截断分支; 同时影响所有后端共享的 `admission` 行为。

```
# python/sglang/srt/managers/schedule_policy.py
# 新增导入: is_gfx95_supported 做硬件门控, lru_cache 保证进程内只判定一次。
from functools import lru_cache
from sglang.srt.utils import get_bool_env_var, is_gfx95_supported, is_hip

@lru_cache(maxsize=1)
def _use_exact_chunk_fill() -> bool:
    """Whether to charge the chunked-prefill compute budget in tokens (gfx95 only).

    Gated on gfx95 because that is where the win is: the aiter absorb bmm picks
    its EVEN_MN specialization from M % BLOCK_SIZE_M, so a short batch costs 2x
    on that kernel, against 1% for the hipBLASLt GEMMs that just run one extra
    partial tile.
    """
    # 双开关: 环境变量默认 True, 但真正的判定看硬件;
    # 非 gfx95 (含 CUDA / NPU) 直接返回 False, 调度行为与 upstream 完全一致。
    return envs.SGLANG_EXACT_CHUNK_FILL.get() and is_gfx95_supported()

class PrefillAdder:
    def __init__(self, ..., dllm_config: Optional[DllmConfig] = None, ...):
        ...
        self.dllm_config = dllm_config
        # dLLM 有独立的 token 预算体系, 这里显式排除, 避免两套计费口径混用。
        self.exact_chunk_fill = _use_exact_chunk_fill() and dllm_config is None

    def _update_prefill_budget(
        self,
        prefix_len: int,
        extend_input_len: int,
        max_new_tokens: int,
        retracted_stain: bool,
        mamba_gap_reserve: int = 0,
        is_chunked_continuation: bool = False,
        compute_charge: Optional[int] = None, # 新增: 计算预算的计费口径
    ):
        """Charge one admitted request against the prefill budgets.
```

```

compute_charge 是计算预算 (rem_chunk_tokens / rem_input_tokens /
rem_dllm_tokens) 的计费值, 单位为 forward 真正会跑的 token; KV 预算
永远按页取整后的 token 计费。默认回退到取整长度, 因此只有
exact-chunk-fill 路径才偏离 upstream 行为。
"""

raw_extend_input_len = extend_input_len
# KV 侧必须按 page 取整: KV pool 的分配粒度就是 page, 少算会 OOM。
extend_input_len = self.ceil_paged_tokens(extend_input_len)
if compute_charge is None:
    compute_charge = extend_input_len

page_overhead = self.page_size
self.rem_total_token_offset += (
    extend_input_len + max_new_tokens + page_overhead + mamba_gap_reserve
)
self.cur_rem_token_offset += extend_input_len + page_overhead + mamba_gap_reserve
if mamba_gap_reserve and self.rem_mamba_slots is not None:
    self.rem_mamba_slots -= 1

# 计算预算改用 compute_charge: 两个计算预算必须同时改, 否则先归零的
# 那个会提前掐断 admission, 把页取整余量留成废预算。
self.rem_input_tokens -= compute_charge

if self.is_hybrid_swa:
    # SWA ring 槽位只在首次准入时预留, 续块再收一次会重复计费。
    if not (self._swa_req_ring and is_chunked_continuation):
        self.rem_swa_token_offset += self._swa_budget_for_req(
            extend_input_len, max_new_tokens
        )

if self.dllm_config is not None:
    self.rem_dllm_tokens -= compute_charge
elif self.rem_chunk_tokens is not None:
    self.rem_chunk_tokens -= compute_charge

# 日志口径统一用 upstream 引入的 raw_extend_input_len, gfx95 路径不变。
self.log_hit_tokens += prefix_len
self.log_input_tokens += raw_extend_input_len

# add_one_req 中的准入与截断: exact-fill 的两处关键改动
def add_one_req(self, req: Req, ...) -> AddReqResult:
    ...
    # 整条装入判定: exact-fill 下用未取整长度比较, 否则一条本来能整装下的
    # prompt 会因为 ceil 后越界而被硬拆成第二段。
    raw_input_tokens = len(req.full_untruncated_fill_ids) - len(req.prefix_indices)
    input_tokens = self.ceil_paged_tokens(raw_input_tokens)
    chunk_fit_tokens = raw_input_tokens if self.exact_chunk_fill else input_tokens

    if ...:

```

```

...
elif chunk_tokens_limit is None or chunk_fit_tokens <= chunk_tokens_limit:
    if (tile_stop := self._check_prefill_tile_budget(input_tokens)) is not None:
        return tile_stop
    # 其余位置参数 (max_new_tokens 等) 保持 upstream 原样, 此处只列受影响的入参
    self._update_prefill_budget(
        prefix_len,
        raw_input_tokens, # 函数内部再 ceil 一次供 KV 预算使用
        ...,
        req.retracted_stain,
        mamba_gap_reserve=mamba_gap_reserve,
        compute_charge=raw_input_tokens if self.exact_chunk_fill else None,
    )
    self._account_prefill_cache_admission(req, prefix_len)
elif self.exact_chunk_fill:
    # 新增分支: 一次取满剩余额度, 让整个 batch 正好命中
    # chunked_prefill_size, dense GEMM 的 M 才能对齐。
    # 用 chunk_tokens_limit 而不是 rem_chunk_tokens, 是为了让 SWA
    # 限流后的 chunk 上限继续生效; 此处 chunk_fit_tokens >
    # chunk_tokens_limit, 所以不会越过 prompt 末尾。
    trunc_len = chunk_tokens_limit
    if trunc_len <= 0:
        return AddReqResult.OTHER

    if truncation_align_size is not None:
        if trunc_len < truncation_align_size:
            return AddReqResult.OTHER
        trunc_len = truncation_align_size * (trunc_len // truncation_align_size)

    if (tile_stop := self._check_prefill_tile_budget(trunc_len)) is not None:
        return tile_stop

    req.set_extend_range(
        len(req.prefix_indices), len(req.prefix_indices) + trunc_len
    )
    self.can_run_list.append(req)
    self.new_chunked_req = req
    self._req_inc_lock_ref(req)
    self._update_prefill_budget(
        prefix_len,
        trunc_len,
        0,
        req.retracted_stain,
        mamba_gap_reserve=mamba_gap_reserve,
        compute_charge=trunc_len, # 逐 token 计费, 无取整损耗
    )
    self._account_prefill_cache_admission(req, prefix_len)

```

python/sglang/srt/envron.py

新增全局开关 `SGLANG_EXACT_CHUNK_FILL = EnvBool(True)`，是 gfx95 上默认开启、其他平台可通过环境变量整体回退的唯一控制点；注释直接指向 `PrefillAdder.exact_chunk_fill`，是理解门控语义的入口。

```
# python/sglang/srt/envron.py
# =====
# Scheduler token budgeting and admission
# =====
SGLANG_PREFILL_TILE_BUDGET_MODE = EnvStr("compact")
SGLANG_PREFILL_DELAYER_MAX_PREFILL_BS_WINDOW_SIZE = EnvInt(16)
# Charge the chunked-prefill compute budget in tokens, not page-ceiled
# tokens, so a prefill batch runs exactly chunked_prefill_size and the dense
# GEMMs get an aligned M. gfx95 only; see PrefillAdder.exact_chunk_fill.
# 默认 True，但真正生效还需 is_gfx95_supported() 为真；
# 在任何平台整体回退到 upstream 行为，设为 False 即可。
SGLANG_EXACT_CHUNK_FILL = EnvBool(True)
```

评论区精华

PR 本身无行内 review 评论，讨论集中在 PR 评论区和一条 APPROVED review 上：

1. 平台覆盖范围 (HaiShaw 的 review) : "@Jacob0226 please extend this to cover gfx942 next, gfx1250 later." 作者回复 "Will cover gfx942 in a follow-up PR, gfx1250 after that."，说明该门控是刻意收窄到 gfx95，收益模型 (absorb bmm 的 EVEN_MN 特化) 不一定在 gfx942 上成立。
2. 跨平台 no-op 的举证方式: HaiShaw 在催 review 时直接替作者背书 "no functional change made to non-gfx/hip"; amd-bot/bingxche 的 CI 报告进一步给出机制解释: "This PR's new exact-chunk-fill logic is gated behind `is_gfx95_supported()` (MI35x only); off gfx95 the `schedule_policy.py` refactor is behavior-preserving (`compute_charge` defaults to the page-ceiled length → identical to upstream), so the NVIDIA/NPU/MLX/MI300 reds cannot come from it." 该报告同时提出两点保留意见: NVIDIA base pipeline 触发 fast-fail 取消了 downstream stage; gfx95 上 3 个 job 因无关原因 red。
3. 与上游 main 的合并冲突处理 (最后一个 commit 92fcd31) : 上游独立引入 `raw_extend_input_len` 修掉了 `log_input_tokens` 的 TODO(lsyin)，作者选择采用上游版本并删掉本分支“仅 gated 路径等价”的实现，理由是 "on every exact-fill call site `raw_extend_input_len` equals `compute_charge`, so the gfx95 path is unchanged" —— 这是本 PR 里最值得注意的一次取舍：优先贴合上游口径，而不是保留自带改动。
4. CI 拉锯: 作者多次 `/rerun-failed-ci` 并贴出 CUDA Base/Extra CI 全绿截图，AMD ROCm 10 的 run 仍是红叉，最终以 `bypass-fastfail` 标签合并。
 - 非 gfx95 平台的 no-op 举证方式 (design): 结论：跨平台等价性由机制（默认值回落）+ 场景重放共同支撑，但不是由仓库内自动化测试保证的。
 - 把同一优化推广到 gfx942 / gfx1250 (design): 已在 review 中确认后续计划，但本次刻意只覆盖 gfx95; gfx942/gfx1250 的收益是否成立 (absorb bmm 的 EVEN_MN 特化路径是否同样存在) 需要各自验证。

- 与上游 raw_extend_input_len 的合并取舍 (design): 采纳上游口径, 避免在公共调度器里长期维护两份等价的指标逻辑; gfx95 行为未变。
- CI 状态与合并门禁 (testing): 在 AMD ROCm 10 CI 未绿、NVIDIA downstream stage 未执行的情况下合并, 等价性结论主要依赖作者自测与 MI35x 作业的间接覆盖。

风险与影响

- 风险:

1. 公共调度路径回归面: schedule_policy.py 是 CUDA/NPU/MLX/ROCm 共享代码, _update_prefill_budget 的签名与计费口径都被改动。作者用 48 场景重放自证开关时逐字节一致, 但这是离线重放, 仓库内没有对应回归测试; 一旦后续别的改动重新定义 extend_input_len 的语义, 这份等价性不会被自动守住。
1. 默认开启的平台级行为变更: SGLANG_EXACT_CHUNK_FILL 默认 True, 因此在 MI355X (gfx95) 上无需任何操作即启用新的截断语义——chunk_tokens_limit 取代 rem_chunk_tokens 决定单次 chunk 长度、且不再 floor 到 page。这是对外可观测的调度行为变更 (trace 中 toks 从 16350 变为 16384), 没有灰度或按模型回退的机制, 只能整体关开关。
2. 与 SWA / tile budget 的交互: 新截断分支在按 truncation_align_size 对齐后调用 _check_prefill_tile_budget(trunc_len), 被拒绝时直接返回 tile_stop, 而不再像旧逻辑那样回退到更小的页对齐长度重试; 在同时开启 SGLANG_PREFILL_TILE_BUDGET (AMD/HIP-only) 的场景下, 这个行为差异没有被测试覆盖。
3. dLLM 路径的边界: exact_chunk_fill 在 dllm_config is not None 时为 False, 但同一份 _update_prefill_budget 现在同时服务两条路径, dLLM 侧只有“compute_charge 默认为取整长度”这一层保护。
4. 精度与 CI 证据强度: GSM8K 0.933 → 0.927 是单次评分, 作者自述在单次分辨率内; AMD ROCm 10 的 CI run 仍失败, NVIDIA base pipeline fast-fail 导致 downstream stage 未执行, 所以“非 gfx95 无影响”缺少完整的端到端绿测背书。
5. 上游耦合风险: 收益的一部分来自同期 ROCm/aiter#4453 (batched_gemm_a8w8 的 gfx950 大 M 配置), PR 明确它“deepens the kernel win but does not gate this PR”, 意味着这两个改动会在不同仓库按不同节奏落地, 排障时需要同时看两侧版本。- 影响: 对用户: MI355X (gfx95) 上的 prefill 吞吐与延迟直接受益。GLM-5.2-MXFP4、TP4、in8192/out1024 下平均 TTFT 改善 5.4% (并发 4) 到 9.5% (并发 64), 中位 TPOT 改善 0.3%–2.9%; 单次 forward 归因显示只有 dense/linear GEMM 显著变化 (173.15 ms → 107.18 ms), sparse-MLA 注意力、all-reduce、MoE、DSA indexer 均在噪声范围内, 说明收益来源与机制解释一致。CUDA、NPU、MLX、MI300 等非 gfx95 用户行为完全不变。

对系统: 预算账本从“单一取整口径”变成“KV 按 page、计算按 token”的双口径, 这是调度器语义层面的结构性变化, 后续任何新增预算维度 (tile budget、mamba gap、SWA ring) 都必须明确自己属于哪一侧。

对团队: HaiShaw 已明确要求把同一思路推广到 gfx942、之后是 gfx1250, 作者承诺 follow-up PR, 因此该门控函数会成为 AMD 各代 GPU 调度优化的公共入口; 同时它也示范了 kernel 侧调优 (aiter 配置) 与调度侧对齐 (本 PR) 可以解耦推进。

- 风险标记：公共调度路径变更，缺少自动化测试覆盖，gfx95 默认开启行为变更，SWA / tile budget 交互未覆盖，CI 未完全绿即合并

关联脉络

- PR #39172 [AMD] gfx950 assembly attention: length-aware split-KV for dynamic workload: 同样面向 gfx950 / MI35X 的 prefill 路径性能优化，把注意力内核的计划改为按长度感知；与本 PR 同属“让 gfx95 上的 M / 序列形状对齐、提高 kernel 特化命中率”的 AMD 性能线。
- PR #39232 [Perf] trtllm_mla: reuse the fused fp8 KV/Q prepare on target verify: 同为 MLA 路径上的 kernel 启动 / 数据准备开销优化（每层 5 次 launch 降为 1 次），与本 PR 一样以“减少与形状对齐相关的前置开销”换取端到端 TTFT、TPOT 收益，是同一类技术权衡的对照样本。
- PR #39360 [AMD] Fix AITER FP8-Q unified-attention Test: 涉及 AMD AITER 注意力后端 (aiter_backend.py) 与 FP8 统一注意力测试，属于本 PR 所依赖的 ROCm/aiter 内核生态同期改动，排查 gfx95 注意力性能问题时需要一并核对版本。