

PR #34432 完整报告

sgl-project/sglang

[AMD][DCP 1/N] add dcp support for aiter backend

合并时间: 2026-09-12 01:35

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/34432>

执行摘要

- 一句话: aiter 后端接入 DCP, Kimi-K3 在 MI355X 跑通 tp8/dcp8
- 推荐动作: 值得精读, 尤其是负责 AMD 注意力后端、DCP/ 长上下文分片或 Kimi-K3 部署的工程师。建议重点看三处设计决策: (1) `_forward_verify_dcp()` 把 target verify 拆成「本 rank 前缀 + 仅 rank 0 计算 draft window + 本地 merge_state_triton」的两段式结构, 以及为绕开 varlen 索引重复而引入的 2-D token table 与 `use_2d_view`; (2) aiter DCP 选择自然对数 LSE 并通过 `is_mla_dcp_lse_base_on_e()` 固化底数约定, 直接影响跨 rank 归约的正确性; (3) head 数记账从单 rank 的 `num_head_padded` 拆成 gather 后的 `mha_kernel_num_head_padded`, 以及随之放宽的断言带来的风险面。同时请把「无自动化测试 + CI 未跑到 DCP 路径」视为后续必须补齐的债务: 任何在此之上继续开发的改动, 都应先补一个能在 CI 中真正执行 `--dcp-size > 1` 的注册测试, 再动这些路径。

功能与动机

PR body 明确说明本补丁是 <https://github.com/sgl-project/sglang/pull/32796> 的 1/N, 目标是“enable dcp support for aiter backend”, 并给出 Kimi-K3 + `--tp-size 8 --dcp-size 8 --prefill-attention-backend aiter --decode-attention-backend aiter --kv-cache-dtype fp8_e4m3 --context-length 132000` 的示例启动命令。长上下文下单卡放不下完整 KV, DCP 把 KV 按 rank 分片, 每个 rank 只需持有约 `seq_len/dcp_size`; 而 aiter 后端此前缺少 LSE 返回、分片索引改写与 verify 两段合并能力, 无法直接复用已有的 DCP 框架。同时 `kimi_k3.py` 的 DCP 分支原本只接受 `cutedsl_mla / tokenspeed_mla`, AMD 平台必须显式放行 aiter。

实现拆解

1. 参数解析与后端选择 (前置门控): `arg_groups/model_overrides/kimi_k3.py` 新增 `_require_kimi_k3_aiter_gluon_dcp_support()`, 在 `_kimi_k3_overrides()` 的 `dcp_size > 1` 分支里加入 `decode_backend == "aiter"` 分支: 先校验 Gluon MLA 内核是否可用, 再把 `prefill` 默认也切成 aiter (仅当用户显式写了 `triton` 才保留), 并把 `AssertionError` 文案扩展为三种后端。`attention_backend_handler.py` 的 `handle_attention_aiter()` 在 DCP 下把 `extend` 直接路由为 `MHA_ONE_SHOT`, 因为该路径会 all-gather 完整序列 KV, 正是 DCP `prefill` 所需; `forward_mla.py` 的 `is_mla_dcp_lse_base_on_e()` 把 aiter 加入集合, 声明 aiter DCP 返回自然对数 LSE, 跨 rank 归约必须用 `exp/log`。

2. aiter 后端的 DCP 元数据层: layers/attention/aiter_backend.py 中 ForwardMetadata 新增 local_kv_lens 与 verify_token_table (每 forward 构建一次而非每层), __init__() 记录 dcp_world_size 与 kv_index_translator; _plan_dcp_decode_metadata() 就地改写 kv_indptr/kv_indices, 把索引翻译成本 rank 分片的物理槽位; _build_dcp_local_kv_lens() 与 _build_dcp_verify_token_table() 分别产出每个请求的分片 KV 长度和 target verify 用的 2-D token table; _get_dcp_graph_max_local_kv_len() 给出 cuda graph 可见的静态上界 ceil(max_context_len / dcp_size); init_cuda_graph_state() 预分配 verify token table 与长度缓冲。head 数记账拆成单 rank 的 num_head_padded 与 gather 后的 mla_kernel_num_head_padded, head 数断言在 dcp_world_size > 1 时放宽。
3. decode 与 target verify 两条计算路径: _forward_decode_dcp() 用 mla_gluon_decode(..., return_lse=True) 只对本 rank 的 KV 分片做注意力并返回自然对数 LSE; _forward_verify_dcp() 分两段执行——stage 1 通过 verify_token_table + use_2d_view=True 读已提交前缀, stage 2 只由 rank 0 执行 (draft window k_window 各 rank 相同), 最后用 merge_state_triton 本地合并两段 LSE。配套地, aiter_mla_gluon.py 的 mla_gluon_decode() 新增 return_lse 与 use_2d_view 两个默认保持旧行为的参数, 并在返回 lse 时保持调用方原有的 flat 输出布局。
4. 模型前向侧接线: models/deepseek_common/attention_forward_methods/forward_mla_rocm.py 增加 force_rope_for_aiter_dcp_decode——DCP 下 aiter 融合 MLA 内核不生效、Gluon 内核不施加 RoPE, 因此 decode / target verify / draft extend v2 必须在 prepare 阶段补 RoPE; forward_absorb_rocm_core() 新增 aiter + DCP 分支: 拼接 q、用 set_mla_kv_buffer() 写 KV (该函数自带 owner 过滤与分片除法, 故 out_cache_loc 原样传入)、target verify 时把 k_window/v_window 交给 attn_mqa_for_dcp_decode() 取回 (attn_output, lse); extend 分支仅在 attn_dcp_metadata 与 dcp_kv_buffer 都存在时才做 all-gather。
5. 横切修正与配套: forward_mha_rocm.py 的 _get_mla_kv_buffer_rocm() 删除 filter_dcp_local_kv_indices() 与 translate_dcp_read_ids(), 避免过滤器与分片除法被执行第二次 (NV CUDA 版本本来就没有这两步); layers/dcp/comm.py 的 all_gather_kv_cache_for_mla_extend() 在 HIP 上前缀长度为 0 时跳过 all-gather, 规避 grid size 0 导致的 invalid configuration; layers/attention/triton_backend.py 为 HIP + DSPARK draft worker 强制 dcp_size=1/dcp_rank=0, 因为 dspark_kv_inject 用原始 cache_loc 写 draft KV, 各 rank 都持有完整副本。测试配套方面本 PR 没有新增任何注册测试 (checklist 中 unit tests 与 documentations 两项未勾选), 准确率数据来自 8xMI355X 上人工执行的 gsm8k 200 评测。

关键文件:

- python/sglang/srt/layers/attention/aiter_backend.py (模块 注意力后端; 类别 source; 类型 core-logic; 符号 _forward_decode_dcp, _forward_verify_dcp, _plan_dcp_decode_metadata, _build_dcp_local_kv_lens): DCP 支持的主体: 新增 _forward_decode_dcp/_forward_verify_dcp 两条分片计算路径、DCP 元数据规划 (_plan_dcp_decode_metadata/_build_dcp_local_kv_lens/_build_dcp_verify_token_table/_get_dcp_graph_max_local_kv_len)、ForwardMetadata 的 local_kv_lens/verify_token_table 字段, 以及 head 数记账与断言放宽, 是整条链路的汇聚

点。

- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py (模块 模型前向; 类别 source; 类型 data-contract; 符号 forward_absorb_rocm_prepare, forward_absorb_rocm_core, force_rope_for_aiter_dcp_decode) : 模型前向侧的接线点: 新增 force_rope_for_aiter_dcp_decode 让 DCP 下的 decode/target verify/draft extend 在 prepare 阶段补 RoPE, 并在 forward_absorb_rocm_core 里新增 aiter + DCP 分支, 用 set_mla_kv_buffer 写 KV、把 k_window 交给 attn_mqa_for_dcp_decode, 是后端新路径真正被调用的地方。
- python/sglang/srt/arg_groups/model_overrides/kimi_k3.py (模块 参数覆盖; 类别 source; 类型 configuration; 符号 _require_kimi_k3_aiter_gluon_dcp_support, _kimi_k3_overrides) : 把 aiter 接入 Kimi-K3 DCP 的参数解析白名单, 并新增 _require_kimi_k3_aiter_gluon_dcp_support() 在参数解析阶段就校验 Gluon MLA 内核是否可用, 避免服务器跑到 kernel launch 才崩溃。
- python/sglang/srt/layers/attention/aiter_mla_gluon.py (模块 注意力内核; 类别 source; 类型 dependency-wiring; 符号 mla_gluon_decode) : mla_gluon_decode 新增 return_lse 与 use_2d_view 两个参数 (默认保持旧行为), 是 DCP 能拿到 LSE 并按行读 KV 的前提; 两个默认值确保非 DCP 路径行为不变。
- python/sglang/srt/layers/dcp/comm.py (模块 集合通信; 类别 source; 类型 dependency-wiring; 符号 all_gather_kv_cache_for_mla_extend) : all_gather_kv_cache_for_mla_extend 在 HIP 上跳过空前缀的 all-gather, 规避 grid size 0 的 invalid configuration 崩溃; 这是平台相关行为分叉, 需要关注。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mha_rocm.py (模块 多头前向; 类别 source; 类型 data-contract; 符号 _get_mla_kv_buffer_rocm) : 删除 _get_mla_kv_buffer_rocm 中重复的 filter_dcp_local_kv_indices 与 translate_dcp_read_ids 调用, 避免过滤与分片除法被执行第二次, 与 CUDA 版本对齐。
- python/sglang/srt/layers/attention/triton_backend.py (模块 后端适配; 类别 source; 类型 core-logic; 符号 TritonAttnBackend.init, is_hip_dspark_draft) : 为 HIP + DSPARK draft worker 强制 dcp_size=1/dcp_rank=0, 避免各 rank 的完整 draft KV 副本被当作分片合并; 带 TODO 说明该逻辑应在非 HIP 平台统一修复, 且只覆盖 DSPARK。
- python/sglang/srt/models/deepseek_common/attention_backend_handler.py (模块 后端选择; 类别 source; 类型 data-contract; 符号 handle_attention_aiter) : handle_attention_aiter 在 DCP 下把 extend 路由为 MHA_ONE_SHOT, 因为该路径会 all-gather 完整序列 KV, 正是 DCP prefill 所需。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla.py (模块 后端判定; 类别 source; 类型 data-contract; 符号 is_mla_dcp_lse_base_on_e) : is_mla_dcp_lse_base_on_e 增加 aiter, 声明 aiter DCP 返回自然对数 LSE, 跨 rank 归约必须用 exp/log 而非 exp2/log2, 是数值正确性的关键契约。

关键符号: _forward_decode_dcp, _forward_verify_dcp, _plan_dcp_decode_metadata, _build_dcp_local_kv_lens, _build_dcp_verify_token_table,

```
_get_dcp_graph_max_local_kv_len, mla_gluon_decode,  
all_gather_kv_cache_for_mla_extend, forward_absorb_rocm_prepare,  
forward_absorb_rocm_core, _get_mla_kv_buffer_rocm,  
_require_kimi_k3_aiter_gluon_dcp_support, _kimi_k3_overrides, handle_attention_aiter,  
is_mla_dcp_lse_base_on_e, TritonAttnBackend.init
```

关键源码片段

python/sglang/srt/layers/attention/aiter_backend.py

DCP 支持的主体：新增 `_forward_decode_dcp/_forward_verify_dcp` 两条分片计算路径、DCP 元数据规划（`_plan_dcp_decode_metadata/_build_dcp_local_kv_lens/_build_dcp_verify_token_table/_get_dcp_graph_max_local_kv_len`）、`ForwardMetadata` 的 `local_kv_lens/verify_token_table` 字段，以及 `head` 数记账与断言放宽，是整条链路的汇聚点。

aiter_backend.py: DCP 下 decode 与 target verify 的核心实现（重新整理后的完整片段）

```
def _get_dcp_graph_max_local_kv_len(self) -> int:  
    """cuda graph 能看到的本 rank 分片长度上界：ceil(max_context_len / dcp_size)。"""  
    w = max(self.dcp_world_size, 1)  
    return (self.max_context_len + w - 1) // w
```

```
def _forward_decode_dcp(self, q, k_buffer, layer, k_descale):  
    """只对本 rank 的 KV 分片做 MLA decode，返回 (out, 自然对数 lse)。
```

跨 rank 的 LSE 合并由调用方负责；这里刻意不复用非 DCP 的 `mla_decode_fwd` 路径，因为那条路径不返回 LSE。

"""

```
fm = self.forward_metadata  
bs = fm.kv_indptr.shape[0] - 1  
num_heads = layer.tp_q_head_num # 已 all-gather，等于 local_heads * dcp_size
```

```
out, lse = mla_gluon_decode(  
    q=q.view(bs, num_heads, layer.qk_head_dim),  
    k_buffer=k_buffer,  
    layer=layer,  
    # kv_indices/kv_indptr 已被 _plan_dcp_decode_metadata 改写成  
    # 「本 rank 分片 + 物理槽位」，这里直接使用  
    kv_indices=fm.kv_indices,  
    kv_indptr=fm.kv_indptr[: bs + 1],  
    sm_scale=layer.scaling,  
    kv_scale=self._resolve_fp8_kv_scale_float(layer, k_descale),  
    min_kv_seq_len=1,  
    return_lse=True, # DCP 必须拿回 LSE，否则无法做跨 rank log-sum-exp 合并  
)  
return out, lse.view(bs, num_heads)
```

```
def _forward_verify_dcp(self, q, k_window, layer, k_descale):
```

"""把已提交前缀与 verify 窗口分两段计算后本地合并 -> (out, 自然对数 lse)。

拆成两段是为了绕开 decode 内核在 DCP 下唯一做不到的事：

按全局位置 $g(j) = j * W + r$ 施加掩码。

"""

```
fm = self.forward_metadata
q_len = fm.max_q_len
num_heads = layer.tp_q_head_num # gathered heads = num_local_heads * dcp
seqused_k = fm.local_kv_lens # 每个请求在本 rank 上的分片 KV 长度
n_rows = seqused_k.shape[0]
bs = n_rows // q_len
```

第一段：本 rank 已提交前缀，用 2-D token table 逐行读取

```
out_a, lse_a = mla_gluon_decode(
    q=q.view(n_rows, num_heads, layer.qk_head_dim),
    k_buffer=self.token_to_kv_pool.get_key_buffer(layer.layer_id),
    layer=layer,
    kv_indices=fm.verify_token_table,
    kv_indptr=seqused_k,
    sm_scale=layer.scaling,
    kv_scale=self._resolve_fp8_kv_scale_float(layer, k_descale),
    min_kv_seq_len=1,
    return_lse=True,
    # varlen 形式会对同一请求的每个 query 重复同一份 index 列表,
    # 所以 verify 必须走接行的 2-D 视图
    use_2d_view=True,
)
lse_a = lse_a.view(n_rows, num_heads)
```

verify 窗口 k_window 由本次 forward 计算且每个 rank 完全相同，

只让 rank 0 参与计算，其余 rank 直接返回前缀部分等待跨 rank 合并

if get_parallel().attn_dcp_rank != 0:

return out_a, lse_a

窗口 latent 按请求优先且连续，它本身就是 KV 池：请求 b 的第 i 行位于

$b * q_len + i$. mla_gluon 在 $seq_len == qlen$ 时的 MTP mask 恰好就是

这里需要的稠密因果窗口，因此可以直接复用同一内核。

```
out_b, lse_b = mla_gluon_decode(
    q=q.view(n_rows, num_heads, layer.qk_head_dim),
    k_buffer=k_window,
    layer=layer,
    kv_indices=torch.arange(n_rows, dtype=torch.int32, device=q.device),
    kv_indptr=torch.arange(bs + 1, dtype=torch.int32, device=q.device) * q_len,
    sm_scale=layer.scaling,
    min_kv_seq_len=1,
    qlen=q_len,
    return_lse=True,
)
```

两段结果都在自然对数域，直接用 triton 版 merge 合并，避免 $\log_2/\exp_2$ 混用

```
return merge_state_triton(out_a, lse_a, out_b, lse_b.view(n_rows, num_heads))
```

python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py

模型前向侧的接线点：新增 `force_rope_for_aiter_dcp_decode` 让 DCP 下的 `decode/target verify/draft extend` 在 `prepare` 阶段补 RoPE，并在 `forward_absorb_rocm_core` 里新增 `aiter + DCP` 分支，用 `set_mla_kv_buffer` 写 KV、把 `k_window` 交给 `attn_mqa_for_dcp_decode`，是后端新路径真正被调用的地方。

`forward_mla_rocm.py`: `aiter + DCP` 的写 KV 与 `decode/verify` 调用路径

```
fuse_rope_for_trtllm_mla = self._fuse_rope_for_trtllm_mla(forward_batch)
```

```
# DCP 下 aiter 融合 MLA 内核不生效，而 Gluon 内核不做 RoPE，  
# 因此 decode / target verify / draft extend v2 必须在这里补上 RoPE，  
# 否则 q_pe/k_pe 是未旋转的原始值，注意力分数整体失真。
```

```
force_rope_for_aiter_dcp_decode = (  
    get_parallel().dcp_enabled  
    and (  
        forward_batch.forward_mode.is_decode()  
        or forward_batch.forward_mode.is_target_verify()  
        or forward_batch.forward_mode.is_draft_extend_v2()  
    )  
    and _use_aiter_gfx95  
    and self.current_attention_backend  
    not in FORWARD_ABSORB_CORE_ATTENTION_BACKENDS  
)  
if self.rotary_emb is not None and (  
    force_rope_for_aiter_dcp_decode  
    or (  
        (not fuse_rope_for_trtllm_mla)  
        and (not self._skip_rope_for_dsa_tilelang_fused())  
        and (not self._skip_rope_for_aiter_fused_mla())  
        and (  
            not _use_aiter  
            or not _is_gfx95_supported  
            or self.use_dsa  
            or (  
                self.current_attention_backend  
                not in FORWARD_ABSORB_CORE_ATTENTION_BACKENDS  
                and self.current_attention_backend != "aiter"  
            )  
        )  
    )  
)  
):  
    q_pe, k_pe = self.rotary_emb(positions, q_pe, k_pe)
```

```
# ... 省略中间逻辑 ...
```

```

elif (
    _use_aiter
    and (
        forward_batch.forward_mode.is_decode()
        or forward_batch.forward_mode.is_target_verify()
        or forward_batch.forward_mode.is_draft_extend_v2()
    )
    and get_parallel().dcp_enabled
):
    q = torch.cat([q_nope_out, q_pe], dim=-1)
    if llama_4_scaling is not None:
        q[..., : self.kv_lora_rank] *= llama_4_scaling
    # set_mla_kv_buffer 内部自己做 owner 过滤与分片除法,
    # 所以这里传原始 out_cache_loc, 不做任何 DCP 改写
    get_token_to_kv_pool().set_mla_kv_buffer(
        self.attn_mqa,
        forward_batch.out_cache_loc,
        k_nope,
        k_pe,
    )
    if forward_batch.forward_mode.is_target_verify():
        # verify 窗口各 rank 内容相同, 作为 k_window/v_window 传入,
        # 由 backend 的 _forward_verify_dcp 决定只让 rank 0 计算
        k_window = torch.cat([k_nope, k_pe], dim=-1)
        v_window = k_nope
    else:
        k_window = None
        v_window = None
    attn_output, lse = self.attn_mqa_for_dcp_decode(
        q,
        k_window,
        v_window,
        forward_batch,
        save_kv_cache=False, # KV 已由上面的 set_mla_kv_buffer 写入
        **(dict(topk_indices=topk_indices) if topk_indices is not None else {}),
    )

```

python/sglang/srt/arg_groups/model_overrides/kimi_k3.py

把 aiter 接入 Kimi-K3 DCP 的参数解析白名单, 并新增 `_require_kimi_k3_aiter_gluon_dcp_support()` 在参数解析阶段就校验 Gluon MLA 内核是否可用, 避免服务器跑到 kernel launch 才崩溃。

kimi_k3.py: aiter + DCP 的参数解析门控 (在 server 启动前就失败, 而不是 kernel launch 时)

```

def _require_kimi_k3_aiter_gluon_dcp_support() -> None:
    from sglang.srt.layers.attention.aiter_mla_gluon import _gluon_fn

```

`_gluon_fn()` 为 None 表示 Gluon MLA 内核不可用 (环境变量关闭、import 失败, # 或 Triton < 3.7 缺少 `cga_layout`), 具体原因已由 `_gluon_fn` 内部打日志。

```

if _gluon_fn() is None:
    raise RuntimeError(
        "Kimi-K3 DCP with decode_attention_backend='aiter' requires the aiter "
        "gluon mla kernel, which is unavailable. See above aborting reasons."
    )

@register_for("KimiK3ForConditionalGeneration")
def _kimi_k3_overrides(server_args: Any, hf_config: Any) -> dict:
    cfg = resolving_view(server_args)
    if cfg.dcp_size > 1:
        overrides = {}
        # ... 省略 symm_mem / DSPARK ragged verify 处理 ...

        prefill_backend, decode_backend = attention_backends_of(cfg)
        if decode_backend == "cuteds_l_mla" or decode_backend is None:
            # ... 省略 cuteds_l_mla 分支 ...
            pass
        elif decode_backend == "tokenspeed_mla":
            # ... 省略 tokenspeed_mla 分支 ...
            pass
        elif decode_backend == "aiter":
            # 先做能力校验，再决定 prefill 后端
            _require_kimi_k3_aiter_gluon_dcp_support()
            # 默认把 prefill 也切到 aiter；只有用户显式指定 triton 时才保留 triton
            prefill_ab = "triton" if prefill_backend == "triton" else "aiter"
            logger.info(
                "Kimi-K3 DCP uses aiter MLA decode: "
                f"prefill={prefill_backend!r} -> {prefill_ab!r}, "
                f"decode={decode_backend!r} -> 'aiter'."
            )
            overrides.update(
                prefill_attention_backend=prefill_ab,
                decode_attention_backend="aiter",
            )
        else:
            raise AssertionError(
                f"Decode attention backend for Kimi-K3 DCP must be 'cuteds_l_mla', 'tokenspeed_mla' or 'aiter', got {decode_backend!r}."
            )
        # ... 省略 dcp_replicate_q_proj / dcp_comm_backend 处理 ...
    return overrides

```

评论区精华

1) DSPARK 覆盖范围：HaiShaw 在 `triton_backend.py:234` 追问 “Only dspark covered?”，作者承认目前只验证了 Kimi-K3-DSPark，另提到存在 `lightseekorg/kimi-k3-eagle3-mla` 但未适配，HaiShaw 回复 “fine, you may do a follow up issue/pr later”，即 EAGLE3 的 DCP

draft 处理留作后续 PR。2) eager_runner.py 的 `forward_batch.extend_prefix_lens` is not `None` 追加检查: HaiShaw 质疑“- bug fix?”, 作者确认这是公共路径上的真实 bug (在 NVIDIA DCP 上也能复现), 并拆出 <https://github.com/sgl-project/sglang/pull/38991> 单独跟踪, 本 PR 内该改动通过 commit 59613d3 “restore eager_runner fix” 还原, 最终 diff 中不再包含 eager_runner.py。3) CI 覆盖争议: amd-bot 给出 “Merge verdict: ❸ Do not merge”, 指出新增的 DCP 代码在 aiter_backend.py 中引入 17 处被禁用的 legacy parallel-getter 调用导致 ratchet 测试失败, 同时强调 “PR CI never exercised the DCP feature this PR adds”——新代码全部被 `dcp_enabled()` (`--dcp-size > 1`) 与 `is_hip()` 门控, AMD pipeline 的 stage-b/stage-c 模型与注意力任务因 `wait-for-stage-a-amd-rocm720` 等待约 4 小时后超时被 fast-fail 跳过。作者修复并多次 `/rerun-failed-ci` 后, HaiShaw 结论为 “CI signal is ok”, review 意见为 “Aiter specific extensions, LGTM.”。4) LSE 底数约定: 从 commit “use natural log” 与 `is_mla_dcp_lse_base_on_e()` 新增 “aiter” 可以看出, aiter DCP 返回自然对数 LSE, 跨 rank 归约必须走 `exp/log` 而非 `exp2/log2`; 这一约定与 `flashmla`、`cutedsl_mla` 保持一致, 否则合并结果会出现系统性数值偏差。

- triton_backend 的 DCP draft 特例目前只覆盖 DSPARK (design): 当前把 HIP + DSPARK draft worker 的 `dcp_size/dcp_rank` 强制为 1/0 的修正只覆盖 DSPARK, EAGLE3 draft 在 HIP + DCP 下的正确性留待后续 PR/Issue; 代码注释也标出 TODO (该逻辑应在非 HIP 平台统一修复)。
- eager_runner 的 `extend_prefix_lens` None 检查是公共路径 bug 修复 (correctness): 确认是公共路径上的真实缺陷 (NVIDIA DCP 同样可复现), 已从本 PR 拆出为独立 PR#38991, 本 PR 最终 diff 不再包含 eager_runner.py。
- PR CI 未真正覆盖新增 DCP 路径, 且 ratchet 测试被新代码触发失败 (testing): 通过修复与重跑恢复了绿灯, 但 DCP 新路径在自动化 CI 中仍缺执行覆盖, 只看 CI 绿灯无法证明该功能可用; 作者仅报告了一个 H20 case 因超时失败。
- aiter DCP 的 LSE 底数约定改为自然对数 (correctness): 采用与既有两个后端一致的自然对数约定, 避免 log 底数不匹配导致的系统性数值偏差; 新后端接入时必须遵守该判定集合。

风险与影响

- 风险: 1) head 数断言被放宽: aiter_backend.py 中 `assert self.dcp_world_size > 1 or _valid_heads or _pad_heads_to_16 or not may_run_mla_decode` 让 DCP 分支无条件通过断言, 实际 head 支持依赖 `mla_kernel_num_head_padded` (gather 后 head 数) 与 Gluon 内核自身上限。参数解析期的检查只覆盖 Kimi-K3 override, 其他 MLA 模型在 aiter + DCP 下可能延迟到 kernel launch 才失败。2) 两段 verify 合并的数值路径: `_forward_verify_dcp()` 依赖 rank 0 之外的 rank 只返回前缀部分, 再由跨 rank LSE 合并补齐; 一旦 `k_window` 在各 rank 不一致 (例如 draft KV 未按预期复制), 会静默产生错误输出而非报错。3) 重复过滤被删除后的契约依赖: `forward_mha_rocm.py` 移除 `filter_dcp_local_kv_indices()` 与 `translate_dcp_read_ids()` 后, 正确性完全依赖三个既有调用方已经预过滤 / 预翻译; 未来新增调用方若忘记预处理, 会读错 KV 槽位且无任何断言保护。4) HIP 专属分支造成平台语义分叉: `dcp/comm.py` 中 `if not _is_hip or dcp_extend_prefix_lens_sum > 0` 只在 HIP 上跳过空前缀 all-gather, CUDA 仍会发起空 collective, 两条路径的可见行为不同且 HIP 侧缺少自动化测试。5) DSPARK-only 的 draft 特例: triton_backend.py 中 `is_hip_dspark_draft` 把 `dcp_size` 强制为 1/`dcp_rank`

为 0，代码自带“TODO: this logic should be fixed in non-hip platform”；EAGLE3 draft 在 HIP + DCP 下仍会把各 rank 的完整 draft KV 副本当作分片合并，属于已知未修复缺陷。6) 长上下文 prefill 的内存压力：handle_attention_aiter() 在 DCP 下把 extend 切到 MHA_ONE_SHOT，该路径会 all-gather 完整序列 KV，132k context 下显存占用与通信量都会显著上升。7) 无自动化测试守护：本 PR 未新增单测 / 注册测试，新增路径又在 CI 中被门控且被 fast-fail 跳过，回归只能靠人工 gsm8k 评测发现。

- 影响：对用户：Kimi-K3 在 8xMI355X (gfx950) 上可选用 aiter 作为 prefill 与 decode 后端并开启 --dcp-size 8，PR 给出 5 类场景的 gsm8k 0.980–0.990、accept len 4.68–4.87，覆盖 dspark、radix cache 与 tp8dcp1 对照。对其他 MLA 模型与 NVIDIA 平台无行为影响（全部被 dcp_enabled() 与 is_hip() 门控）；但对 HIP + DCP + EAGLE3 draft 的组合仍是未支持状态。对系统：DCP 的分片索引改写与 LSE 合并语义开始进入 aiter 后端，ForwardMetadata 增加了每 forward 构建一次的 DCP 字段，属于注意力子系统的结构性扩展；LSE 底数约定 (is_mla_dcp_lse_base_on_e) 成为跨后端共享契约，后续新后端接入必须遵守。对团队：作为 1/N 系列的开篇，后续至少还有 EAGLE3 draft 适配、非 HIP 平台通用化、以及拆出的 PR#38991 (eager_runner extend_prefix_lens bug) 三条明确跟进线，维护者需持续关注 CI 是否真正覆盖 DCP 路径。
- 风险标记：核心注意力路径变更，缺少测试覆盖，DCP 路径 CI 未执行，head 数断言放宽，平台语义分叉（HIP 专属分支），EAGLE3 draft 未适配，重复过滤删除后的契约依赖

关联脉络

- PR #32796 [AMD][DCP] original series enabling DCP on aiter backend: PR body 明确说明本 PR 是 #32796 的 1/N 拆分，速度与性能结论也直接引用该 PR。
- PR #38991 split-out fix for eager_runner extend_prefix_lens None check: review 中 HaiShaw 追问的 eager_runner 公共路径 bug 被拆分为独立 PR，作者在本 PR 评论中给出链接并从本 PR 还原该改动。
- PR #38755 [AMD] aiter: fail loudly on cross-layer KV sharing in target_verify: 同样修改 python/sglang/srt/layers/attention/aiter_backend.py 的 target_verify 路径，与本 PR 的 _forward_verify_dcp 处于同一代码区域，存在潜在冲突与语义重叠。
- PR #38993 [Refactor] Generalize attention graph variants in the decode runner: 同为注意力后端 / 解码图变体的改造，本 PR 新增了 DCP 专用的 cuda graph 状态 (init_cuda_graph_state、_get_dcp_graph_max_local_kv_len)，两者在 graph 捕获契约上相互关联。