

PR #39648 完整报告

sgl-project/sglang

dsv4.1: Top-k kernels and candidate selection

合并时间: 2026-09-16 14:54

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39648>

执行摘要

- 一句话: DSV4.1 新增索引器内核, 重做 Top-K v2 分派
- 推荐动作: 建议精读, 重点关注三处: topk.py 的探测驱动 JIT 宏与 plan_topk_v2 默认值语义、topk_impl.cuh 的 NaN padding 与 coarse_bin_lower_bound 的 round-to-even 补偿、topk_v2.cuh 的 PageTransform 解耦与 cluster 8/16 分派。四个值得学习的设计决策: 用 NaN 而非极小值填充以保护合法的 -inf; 用 ULP 步进而非重算 fp16 bin 处理 round-to-nearest-even; 用 cudaOccupancyMaxActiveClusters 而不是 SM 数除法定 pool 尺寸 (GPC 约束); 把 page-table 变换从选择内核解耦成独立结构以便两种输出模式共用。若负责索引器或 Blackwell 性能, 建议先补齐 BBuf 要求的边界 benchmark 再信任这些阈值, 并为三个新内核补注册测试。

功能与动机

PR body 明确目标是补齐 DeepSeek-V4.1 稀疏索引器所需的算子: "Add the DeepSeek-V4.1 sparse-indexer kernels that sit beside Top-k v2: a bf16 small-row top-k, the two-level indexer's block-max keys and sorted block table, and a cluster-occupancy probe."。另一条动机是修 Top-K v2 自身的缺陷: padding 用 -FLT_MAX 时会把合法的 -inf 分数挤出 top-k (PR 新增 test_topk_v2_negative_infinity_ties、test_topk_v2_ragged_negative_infinity 两个回归用例钉住该行为); coarse_bin_lower_bound 在 fp16 尾数为奇数的 bin 上因 fp32 -> fp16 的 round-to-nearest-even 取到偏低阈值; cluster 路径此前固定宽度 8、persistent pool 靠手算 SM 比例, 本次改为探测驱动。改动前的 docstring 还记录过 GLM 5.2 MTP 的 DP-idle companion 行负长度会毒化 plan 并触发非法访存, 本 PR 把它收敛成 "seq_lens 必须非负、0 表示无 token" 的契约说明。

实现拆解

1. 入口层拆分: Top-K 与索引器 helper 分家

- python/sglang/kernels/ops/attention/dsv4/topk.py 只保留 top-k 入口, 新增 _jit_topk_bf16_small_module / topk_transform_bf16_small, 并把 _jit_topk_v2_module 的 cuda_wrappers 改成模板化写法 TopKKernel<args>::transform_paged|transform_ragged|plan。
- 两级索引器的两个 helper 移到新文件 python/sglang/kernels/ops/attention/dsv4/candidate_table.py: amax8_varlen (一级 block key) 与 sort_candidate_blocks (排序后的 block table), 各自用 @cache_once 缓存 JIT 模块, 并按 is_arch_support_pdl() 传编译

期参数。这样模型侧可以独立引用索引器算子，不再混装在 topk.py 里。

2. 三个新 CUDA 内核

文件	内核 / 入口	关键设计
jit/csrc/deepseek_v4/topk_bf16_small.cuh (+440)	TopKBF16Kernel::transform / topk_transform_bf16_small	改自 DeepSelect 的 init-window select：一行一个 CTA (512 线程、每线程 32 个 score 常驻寄存器)，两趟 radix 扫原始 bf16 字节精确定位第 k 大，census 后经 shared memory 一次性做 page-table 变换写出；直方图按 raw byte 索引、用 kNegShift 把正负半区错开避免 bank 冲突；契约是行长 ≤ 16384 、 $k \leq 2048$ 、不支持 NaN 分数
jit/csrc/deepseek_v4/block_amax.cuh (+155)	BlockAmaxKernel::amax8_varlen	每 8 个 score 取 max 作为一级 key，行内最新块强制写 +inf（新块必被选中）；块数不超过 topk 的短行整体跳过；要求 fp32、scores 行 32 B 对齐，并显式用 fmaxf 而非 amax 语义（NaN 被忽略）
jit/csrc/deepseek_v4/candidate_block_table.cuh (+218)	CandidateBlockTableKernel::transform / sort_candidate_blocks	对已选 block 做 bitmap 计数排序：只含 1 位的词由持有线程直接写出，密集词进 block 级队列、由各 warp 一线程一位 drain；同一趟把 block id 映射为 pool slot ($\text{page_table}[b, \text{id} / \text{bpp}] * \text{bpp} + \text{id} \% \text{bpp}$)； $\text{nblocks} \leq \text{topk}$ 的行走恒等表并跳过读取输入

3. 占用率探测决定 JIT 期尺寸

- 新增 python/sglang/kernels/jit/utils/occupancy.py 与 jit/csrc/occupancy/cluster_probe.cuh (55 行探针)：get_max_active_clusters(cluster_size, occupancy) 通过 cudaOccupancyMaxActiveClusters 询问驱动，而不是用 SM 数除以 cluster 宽度（cluster 需要落在同一个 GPC 内，除法会高估）；探针用 shared memory 预算把 block 钉在指定 occupancy 上，返回 0 时抛 ValueError。
- _jit_topk_v2_module 用探测结果生成 -DSGL_TOPK_V2_MAX_C8_OCC2 与 -DSGL_TOPK_V2_MAX_C16_OCC1，分别决定 persistent pool 尺寸与 cluster-16 小批量上限；探测失败或结果非正时不下发宏，交由 topk_v2.cuh 的按 arch 默认值兜底（注释明确：传 0 会把 persistent pool 缩成空网格）。

4. Top-K v2 边界与分派重做

- padding 语义：topk_impl.cuh 新增 padding_value() 返回 quiet NaN（替代 -FLT_MAX），padding bin 由 extract_coarse_bin(padding_value()) 推导，因此直方图校正不依赖平台的 NaN 编码；extract_coarse_bin 改为在 fp16 位模式上做符号有序化，coarse_bin_lower_bound 在 fp16 尾数为奇数的 bin 上把中点抬一个 ULP。

- cluster 路径: TopKCluster<N> 支持 N = 8 与 16, 输入在 rank 间交错, rank 0 独占直方图与阈值并用 release/acquire cluster barrier 广播; topk_plan_cluster 携带每 arch 候选表; SUPPORT_CLUSTER 收窄到 sm90/sm100, sm120 与 ROCm 走 streaming 路径。同时移除 #32830 引入的 __builtin_assume 缓解 (重构后的 epilogue 在 nvcc 13.1–13.4 / sm_90a 下不再让 cicc 段错误), TieHandleSmem 不再过度对齐计数器。
- plan_topk_v2(static_threshold) 默认值由 0 改为 -1: 负数才走内核启发式, 0 现在表示字面阈值。
- page-table 变换从 TopKProblem 解耦为 PageTransform (含 page_to_indices), trivial_transform / 新的 paged_transform 接收它; get_transform(batch_id) 对可选指针保留 nullptr 分支 (review 修复项)。

5. 测试与工具配套

- test/registered/kernels/ops/attention/test_topk_v2.py: 新增 _assert_topk_values (校验数量、范围、唯一性并与 torch.topk 值集合比对) 与两个 -inf 回归; FIXED_CONFIGS 去掉写死的 FLOOR = 65536, 改为跨过 arch 相关边界; 越界写断言在 review 后恢复。测试注册为 CUDA base-b-kernel-unit (1-gpu-large) 与 AMD jit-kernel-unit。
- python/sglang/test/kits/dsa_metadata_kit.py 改用公开的 topk_v2_plan_is_written(seq_lens) 判断 plan 是否被写过, 取代对私有 JIT module 的引用, 降低测试对实现细节的耦合。
- test/registered/kernels/benchmark/attention/bench_topk.py 为每个配置加随机种子并给 paged sweep 增加 page_size 维度, 但 PR body 声明本次未重跑。

关键文件:

- python/sglang/kernels/ops/attention/dsv4/topk.py (模块 Top-K 内核; 类别 source; 类型 core-logic; 符号 _jit_topk_bf16_small_module, topk_transform_bf16_small, _jit_topk_v2_module, plan_topk_v2): Top-K 入口层: 新增 bf16 小行 Top-K 入口, 用占用率探测结果生成 JIT 宏决定 persistent pool 尺寸, 改 plan_topk_v2 默认语义并新增公开的 topk_v2_plan_is_written 探针。
- python/sglang/kernels/jit/utils/occupancy.py (模块 占用率探测; 类别 source; 类型 infrastructure; 符号 _jit_probe_module, _get_max_active_clusters, get_max_active_clusters): 新增的占用率探测 API, 是 cluster 路径从 " 手算 SM 比例 " 转向 " 问驱动 " 的关键: persistent pool 尺寸与 cluster-16 上线都由它决定, 返回 0 时显式报错。
- python/sglang/kernels/ops/attention/dsv4/candidate_table.py (模块 索引器; 类别 source; 类型 infrastructure; 符号 _jit_block_amax_module, amax8_varlen, _jit_candidate_block_table_module, sort_candidate_blocks): DeepSeek-V4.1 两级索引器的 host 侧入口: 一级 block max 键 (amax8_varlen) 与候选块表排序 (sort_candidate_blocks), 是从 topk.py 拆出的新模块。
- python/sglang/kernels/jit/csrc/deepseek_v4/topk_v2.cuh (模块 Top-K 内核; 类别 source; 类型 core-logic; 符号 PageTransform, get_transform, page_to_indices, trivial_transform): Top-K v2 的分派与数据结构核心: 新增 PageTransform 解耦 page-table 变换、TopKCluster<8>/<16> 交错输入与 cluster barrier、per-arch cluster

候选表，并把 cluster floor 相关常量改成运行期 / 探测驱动。

- python/sglang/kernels/jit/csrc/deepseek_v4/topk_impl.cuh (模块 Top-K 内核; 类别 source; 类型 core-logic; 符号 padding_value, infinity_value, extract_coarse_bin, coarse_bin_lower_bound) : 边界语义的修复点: padding_value 改 NaN、padding bin 由它推导、coarse_bin_lower_bound 处理 round-to-nearest-even 与 key 空间边缘、SUPPORT_CLUSTER 收窄平台范围、MaxSmem 简化。
- python/sglang/kernels/jit/csrc/deepseek_v4/topk_bf16_small.cuh (模块 候选筛选; 类别 source; 类型 core-logic; 符号 TopKBF16Config, TopKBF16Params, topk_bf16_small_kernel, topk_bf16_find_pivot_warp) : 新增 440 行的 bf16 小行 Top-K 内核, 改自 DeepSelect 的 init-window select, 是 DSV4.1 消费行的核心算子; 本 PR 未给它注册测试。
- python/sglang/kernels/jit/csrc/deepseek_v4/candidate_block_table.cuh (模块 索引器; 类别 source; 类型 core-logic; 符号 CandidateBlockTableConfig, CandidateBlockTableParams, sort_128k_transform, CandidateBlockTableKernel::transform) : 新增候选块表内核: 对已选 block 做 bitmap 计数排序并顺带完成 pool slot 映射, 是两级索引器第二级的落地点。
- python/sglang/kernels/jit/csrc/deepseek_v4/block_amax.cuh (模块 索引器; 类别 source; 类型 core-logic; 符号 BlockAmaxConfig, BlockAmaxParams, amax8_varlen_kernel, BlockAmaxKernel::amax8_varlen) : 新增一级 block max 键内核 : 每 8 个 score 取 max 作为一级 key, 最新块强制 +inf, 短行整体跳过, 是两级索引器的第一级。
- test/registered/kernels/ops/attention/test_topk_v2.py (模块 内核测试; 类别 test; 类型 test-coverage; 符号 _assert_topk_values, test_topk_v2_negative_infinity_ties, test_topk_v2_ragged_negative_infinity, FIXED_CONFIGS) : Top-K v2 的回归护栏: 新增 _assert_topk_values 与两个 -inf 回归用例, 去掉写死的 cluster floor, 并在 review 后恢复了越界写断言。
- python/sglang/test/kits/dsa_metadata_kit.py (模块 元数据夹具; 类别 test; 类型 test-coverage; 符号 assert_metadata_equal) : 测试 kit 不再引用私有 JIT module, 改用公开的 topk_v2_plan_is_written 判断 plan 是否写出, 降低测试对实现细节的耦合。

关键符号: topk_transform_bf16_small, amax8_varlen, sort_candidate_blocks, get_max_active_clusters, _jit_topk_v2_module, plan_topk_v2, topk_v2_plan_is_written, get_transform, page_to_indices, padding_value, coarse_bin_lower_bound, _assert_topk_values

关键源码片段

python/sglang/kernels/ops/attention/dsv4/topk.py

Top-K 入口层: 新增 bf16 小行 Top-K 入口, 用占用率探测结果生成 JIT 宏决定 persistent pool 尺寸, 改 `plan_topk_v2` 默认语义并新增公开的 `topk_v2_plan_is_written` 探针。

```
# python/sglang/kernels/ops/attention/dsv4/topk.py (新内核入口与 plan 语义节选)
```

```
@cache_once
```

```
def _jit_topk_bf16_small_module():
    # 与 Top-K v2 相同的 PDL 能力开关, 编译出平台特化版本
    args = make_cpp_args(is_arch_support_pdl())
    return load_jit(
        make_name("topk_bf16_small"),
        *args,
        cuda_files=["deepseek_v4/topk_bf16_small.cuh"],
        cuda_wrappers=[("topk_transform", f"TopKBF16Kernel<{args}>::transform")],
    )
```

```
def topk_transform_bf16_small(
    scores: torch.Tensor,
    seq_lens: torch.Tensor,
    page_table: torch.Tensor,
    out_page_indices: torch.Tensor,
    page_size: int,
) -> None:
    """bf16 top-k for rows of at most 16384 scores (the DeepSeek-V4.1 sparse
    indexer's consumer rows), fused with a page-table transform.

    Row ``b`` selects the ``k = out_page_indices.shape[1]`` best of its first
    ``seq_lens[b]`` scores (``k`` at most 2048); a selected index ``i`` is
    written as ``page_table[b, i // page_size] * page_size + i % page_size``,
    in no particular order, and ``-1`` fills the slots past
    ``min(k, seq_lens[b])``. Selection is exact (two radix passes over the raw
    bf16 bytes locate the k-th largest value); which of the elements equal to
    it fill the last slots is arbitrary. NaN scores are not supported.
    """
    # 契约: 行长 <= 16384、k <= 2048、不支持 NaN 分数, 越界即错误结果
    _jit_topk_bf16_small_module().topk_transform(
        scores, seq_lens, page_table, out_page_indices, page_size
    )
```

_PLAN_METADATA_INTS_PER_BATCH = 2

```
def plan_topk_v2(seq_lens: torch.Tensor, static_threshold: int = -1) -> torch.Tensor:
    """
    Preprocess the per-batch routing plan for :func:`topk_transform_paged_v2`.
    NOTE: every entry of ``seq_lens`` must be NON-NEGATIVE.

    :param static_threshold: If a batch item has ``seq_len`` > ``static_threshold``,
        prefer the cluster implementation.
        Negative number means internal heuristic.
    """
    # 默认值由 0 改为 -1: 负数走内核启发式, 0 及以上是字面阈值
    module = _jit_topk_v2_module()
```

```

bs = seq_lens.shape[0]
# ... (其余: 分配 metadata 并调用 module.topk_plan(seq_lens, metadata, static_threshold))
return metadata

```

```

def topk_v2_plan_is_written(seq_lens: torch.Tensor) -> bool:
    """Whether :func:`plan_topk_v2` writes a plan for these lengths. Small
    batches and devices without clusters leave the plan buffer untouched."""
    # 多留一行哨兵: 若第 0 行第 1 列仍是 -1, 说明这次 plan 根本没写
    probe = torch.full(
        (seq_lens.shape[0] + 1, _PLAN_METADATA_INTS_PER_BATCH),
        -1,
        dtype=torch.int32,
        device=seq_lens.device,
    )
    _jit_topk_v2_module().topk_plan(seq_lens, probe, -1)
    return probe[0, 1].item() != -1

```

python/sglang/kernels/jit/utils/occupancy.py

新增的占用率探测 API, 是 cluster 路径从 " 手算 SM 比例 " 转向 " 问驱动 " 的关键:
persistent pool 尺寸与 cluster-16 上线都由它决定, 返回 0 时显式报错。

```

# python/sglang/kernels/jit/utils/occupancy.py (完整实现)
"""Occupancy probes a host-side dispatch needs before it can size a grid."""

from __future__ import annotations

from typing import TYPE_CHECKING

from sglang.kernels.jit.utils.common import cache_once
from sglang.kernels.jit.utils.compile import load_jit

if TYPE_CHECKING:
    from tvm_ffi.module import Module

__all__ = ["get_max_active_clusters"]

@cache_once
def _jit_probe_module() -> Module:
    # 探针内核只有 55 行, 编译一次后由 cache_once 常驻
    return load_jit(
        "occupancy_cluster_probe",
        cuda_files=["occupancy/cluster_probe.cuh"],
        cuda_wrappers=[("get_max_active_clusters", "get_max_active_clusters")],
    )

@cache_once

```

```

def _get_max_active_clusters(cluster_size: int, occupancy: int) -> int:
    # 同一组 (cluster_size, occupancy) 只问驱动一次
    return int(_jit_probe_module().get_max_active_clusters(cluster_size, occupancy))

def get_max_active_clusters(cluster_size: int, occupancy: int) -> int:
    """Clusters of ``cluster_size`` blocks that can be resident at once.

    Asks the driver (``cudaOccupancyMaxActiveClusters``) rather than dividing SM
    count by cluster size: a cluster's blocks must be co-scheduled within one
    GPC, so the answer falls short of the division once the cluster stops
    dividing a GPC evenly. The probe kernel is pinned to ``occupancy`` blocks per
    SM, so pass the occupancy the real kernel reaches (its second
    ``__launch_bounds__`` argument). Raises ``RuntimeError`` before sm90, which
    has no clusters, and ``ValueError`` when nothing is schedulable.
    """
    result = _get_max_active_clusters(cluster_size, occupancy)
    if result == 0:
        # 返回 0 会把 persistent pool 缩成空网格；宁可显式失败，让调用方回退到 arch 默认值
        raise ValueError(
            f"no cluster of {cluster_size} fits at occupancy {occupancy}; "
            "the cluster width is likely beyond what this device supports"
        )
    return result

```

python/sglang/kernels/ops/attention/dsv4/candidate_table.py

DeepSeek-V4.1 两级索引器的 host 侧入口：一级 block max 键（amax8_varlen）与候选块表排序（sort_candidate_blocks），是从 topk.py 拆出的新模块。

python/sglang/kernels/ops/attention/dsv4/candidate_table.py（两级索引器的两个入口）

```

@cache_once
def _jit_block_amax_module():
    # PDL 能力随 arch 变化，因此编译期模板参数也要跟着变
    args = make_cpp_args(is_arch_support_pdl())
    return load_jit(
        make_name("block_amax"),
        *args,
        cuda_files=["deepseek_v4/block_amax.cuh"],
        cuda_wrappers=[("amax8_varlen", f"BlockAmaxKernel<{args}>::amax8_varlen")],
    )

def amax8_varlen(
    scores: torch.Tensor,
    seq_lens: torch.Tensor,
    topk: int = 0,
    *,
    max_seqlen: int = 0,

```

```

    out: Optional[torch.Tensor] = None,
) -> torch.Tensor:
    """Level-one keys of the two-level indexer: ``out[b, i]`` is the max of
    ``scores[b, 8 * i : 8 * i + 8]`` for ``i < ceil(seq_lens[b] / 8)``; the last of
    them ``+inf`` (the newest block is always selected), nothing written past
    that count. Rows with at most ``topk`` blocks are skipped (every block is
    selected anyway); ``topk=0`` never skips. ``out`` is allocated as
    ``[rows, ceil(max_seqlen / 8)]`` when not given, ``max_seqlen`` defaulting to
    the width of ``scores``; every ``seq_lens[b]`` must fit in ``8 * out.shape[1]``.
    fp32 only for now; ``scores`` rows must be 32-byte aligned (stride a multiple
    of 8). Returns ``out``.
    """
    if out is None:
        num_tokens, max_len = scores.shape
        if max_seqlen == 0:
            max_seqlen = max_len
        # 每个 8 个 score 压缩成一个 key; 最新块由内核强制写成 +inf
        out = scores.new_empty(num_tokens, (max_seqlen + 7) // 8)
        _jit_block_amax_module().amax8_varlen(scores, seq_lens, out, topk)
    return out

```

@cache_once

```

def _jit_candidate_block_table_module():
    args = make_cpp_args(is_arch_support_pdl())
    return load_jit(
        make_name("candidate_block_table"),
        *args,
        cuda_files=["deepseek_v4/candidate_block_table.cuh"],
        cuda_wrappers=[("transform", f"CandidateBlockTableKernel<{args}>::transform")],
    )

```

```

def sort_candidate_blocks(
    blocks: torch.Tensor,
    seq_lens: torch.Tensor,
    page_table: torch.Tensor,
    page_size: int,
    *,
    out_pages: Optional[torch.Tensor] = None,
) -> torch.Tensor:
    """The block table of the two-level indexer from a row's selected blocks,
    in place: ``blocks`` ``[rows, k]`` int32 block ids in any order, ``-1``
    padded, become the same ids ascending with ``INT32_MAX`` past ``min(k,
    ceil(seq_lens[b] / 8))``; the matching pool slots / 8 (``page_table[b, id //
    bpp] * bpp + id % bpp``, ``bpp = page_size // 8``, same padding) go to
    ``out_pages``. A row with at most ``k`` blocks gets the identity table
    regardless of its input. Returns ``out_pages``.
    """

```

```

if out_pages is None:
    out_pages = torch.empty_like(blocks)
# blocks 被就地改写为升序；out_pages 需要调用方保证与 blocks 同形
_jit_candidate_block_table_module().transform(
    blocks, seq_lens, page_table, out_pages, page_size
)
return out_pages

```

评论区精华

- 性能证据缺口（未解决）：BBuf 在总体 review 中写道："This changes occupancy probing, cluster-size selection, persistent dispatch, and several plan thresholds, but the PR description says the benchmark was not run."，要求补充 plan 选择与延迟数据，覆盖 batch 15/16 与序列长度 24K/30K/32K/64K 附近，在目标 Blackwell 与 H100 上给出 baseline/candidate commit 与命令；还指出 review 中 "14 files are byte-identical" 的表述在 fd6e5f8 之后需要更新。材料中未见作者对这条的后续答复，PR body 也承认 bench_topk.py "not re-run here"。
- nullptr 指针算术（已修复）：BBuf 在 topk_v2.cuh 的 get_transform() 上指出："Both pointers are optional depending on the output mode, but get_transform() unconditionally performs pointer arithmetic on them... nullptr + offset is undefined behavior even if the corresponding field is later eliminated by an if constexpr. Could we preserve the null value here?"，并给出三元表达式的修法。作者回复 "Fixed in 263375f167: both optional pointers now preserve nullptr instead of performing pointer arithmetic on it."，最终 head 代码确实带 == nullptr ? nullptr : 分支。
- 越界写断言（已恢复）：BBuf 在 test_topk_v2.py 上要求保留原断言："It verified that the only legal in-place mutation was the at-most-three aligned prefix columns before a window... an out-of-window write can now pass all tests as long as the selected Top-k values remain correct."；作者恢复该断言并保留新的 -inf 覆盖，称集成测试在 H200 上 288/288 通过。
- CI 补跑：初始 CI 汇总中 Base / Extra / AMD 三项都是未通过 (no_entry_sign / x) 状态，作者用 /rerun-test test_topk_v2.py 以及 dsa_indexer、dsa_metadata_replay、dsv32_indexer_fusion、dsv4_nonpaged_indexer 一组用例补跑后，1-gpu-h100 与 ubuntu-latest 均转绿。
 - 可选指针的 nullptr 算术 (undefined behavior) (correctness): 作者回复已在 263375f167 修复；最终 head 的 get_transform 保留了 == nullptr ? nullptr : 分支。
 - 删除越界写断言导致就地改写缺少护栏 (testing): 作者恢复断言并保留新的 -inf 覆盖，注明保留的集成测试在 H200 上 288/288 通过。
 - 分派阈值与占用率改动缺少 benchmark 证据 (performance): 材料中未见作者对该请求的答复，PR body 仍写明 bench_topk.py "not re-run here"，属于未闭合事项。
 - PR 描述与最新 diff 不一致 (documentation): 材料中未见更新后的描述文本，属于遗留的文档一致性问题。

风险与影响

- 风险:

- 新增内核零注册测试: `topk_transform_bf16_small`、`amax8_varlen`、`sort_candidate_blocks` 三个内核没有自己的注册测试, PR body 说明仅与 torch reference 对比过、"covered by the model's end-to-end tests"; 若 e2e 覆盖不到 $k = 2048$ 、`page_size` 非 64、非 PDL 或大 `page_size` 等组合, 回归会静默。
- 默认参数语义变更: `plan_topk_v2` 的 `static_threshold` 默认值从 0 变为 -1, 0 的含义由 "启发式" 变成 "字面阈值 0" (任何非空行都倾向 cluster)。PR 称树内无调用方传该参数, 但下游 / 实验分支若显式传 0, 分派行为会改变。
- 平台分派收窄: `SUPPORT_CLUSTER` 限定 sm90/sm100, sm120 与 ROCm 从 cluster 退回 streaming; 长上下文、 $\text{batch} \leq 512$ 场景在 sm120 上的性能需要重新确认。
- 分派阈值缺乏基准证据: cluster floor、persistent pool 尺寸、cluster-16 小批量上限现在来自探测 + 经验默认值, BBuf 要求的 batch 15/16、seq 24K/30K/32K/64K 边界数据缺失, 阈值是否合理无法从材料判断。
- 探测失败的静默回退: `_jit_topk_v2_module` 用 `except Exception: pass` 吞掉探测异常并回落到 `topk_v2.cuh` 的按 arch 默认值; 若驱动或架构上探测持续失败, 实际 pool 尺寸与预期不一致且只体现为性能差异, 不易察觉。
- 新内核的隐含契约: `topk_transform_bf16_small` 要求行长 ≤ 16384 、 $k \leq 2048$ 且不支持 NaN 分数; `amax8_varlen` 要求 fp32、行 32 B 对齐且 $\text{seq_lens}[b] \leq 8 * \text{out.shape}[1]$ (注释明说不做检查); `sort_candidate_blocks` 要求 `page_size` 是 ≥ 8 的 2 的幂。调用方越界即产生错误结果或 UB。
- NaN padding 的语义依赖: `padding bin` 由 `extract_coarse_bin(padding_value())` 推导, 避免依赖平台 NaN 编码, 但该假设本身需要在不同架构 (含 ROCm) 上验证; 新增回归只覆盖 -inf tie, 不覆盖 NaN 编码差异。

- 影响:

- 用户: 使用 DeepSeek-V4.1 稀疏索引器 / 长上下文的推理请求会走到新内核; 含 -inf 分数的输入 (mask 位置、DP-idle 行) 在 Top-K v2 下结果更正确, 不再被 -FLT_MAX 填充挤掉。
- 系统: JIT 首次编译多两次驱动占用率探测与额外 -D 宏, 模块缓存键随设备 arch/ 探测结果变化, 跨设备部署会各编译一份; `cache_once` 之后运行期开销仅为一次查表。
- 平台与 CI: ROCm 与 sm120 的 Top-K 分派路径改变 (AMD 的 `jit-kernel-unit` 用例已注册); DSA 相关 kit 改用公开 API, 减少测试对私有模块的耦合。
- 团队 / 维护: 索引器算子从 `topk.py` 拆到 `candidate_table.py`, 模块边界更清晰, 但新内核缺少注册测试意味着后续重构缺少护栏。
- 风险标记: 核心路径变更, 新内核缺少注册测试, 基准数据未验证, 默认参数语义变更, 平台分派收窄, 探测失败静默回退

关联脉络

- PR #39677 dsv4.1: Rust extension modules for image preprocessing, KV pool names, and PD bootstrap: 同属 DeepSeek-V4.1 的迭代线 (本 PR 分支名即 dsv4.1-topk), 一个补 Rust 侧布局、一个补 JIT 内核与候选选择, 共同构成 dsv4.1 的功能拼图。

- PR #36534 [DSV4][BCG] Optimize the heavy memory use of C4 Indexer when BCG is enabled: 同属 DSV4 indexer 方向的前序工作 (C4 indexer 与 prefill 图内存优化) ; 本 PR 的 -inf padding 修复与 cluster 分派调整作用在同一 indexer 路径上。
- PR #39155 [AMD] GLM-5.2 NextN: cast draft fused MoE to per-channel FP8: Top-K v2 的 docstring 记录过 GLM 5.2 MTP 的 DP-idle companion 行负长度会毒化 plan 并触发非法访存; 本 PR 把该历史教训收敛为非负契约说明, 并为同类 -inf/-1 边界新增回归。
- PR #32830 (上下文未提供标题; PR body 引用的 __builtin_assume 缓解来源): 本 PR 移除了 #32830 在小批量 cluster 尾声引入的 __builtin_assume 缓解, 理由是重构后的 epilogue 在 nvcc 13.1–13.4 / sm_90a 下不再让 cicc 段错误。