PR #34696 完整报告

sgl-project/sglang

[Spec] Support logprobs with DSpark speculative decoding

合并时间：2026-08-17 06:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34696

执行摘要

- 一句话：DSpark 推测解码支持 logprobs 响应
- 推荐动作：值得精读，尤其是 spec 算法与 logprob 计算的 wiring 模式。重点关注三点：一是 compute_spec_v2_logprobs + 线性接受索引的对齐手法，后续新 spec 算法可直接复用；二是 _linear_accept_indices 的缓存设计，避免热路径重复分配；三是 " 先拒绝后解锁 " 的最小改动面——两处拒绝逻辑 + 一处热路径接入就完成能力开放。建议阅读时同时对照 test_basic_sanity_dspark.py 的 compact 模式环境变量，理解测试如何覆盖折叠布局。

功能与动机

DSpark（DeepSeek 的块级推测解码）此前在 SGLang 中明确拒绝 return_logprob：scheduler.py 的 handle_generate_request 与 dspark_worker_v2.py 的 forward_batch_generation 各有一处 "DSpark speculative decoding does not support return_logprob yet." 硬拦截。PR body 的目标是 "Enable OpenAI-compatible logprob responses when serving with DSpark speculative decoding"。技术难点在于 DSpark 的 verify/accept 输出是折叠布局（含 compact ragged-verify 模式），被接受 token 与 target logits 行并不天然一一对应，必须有一套索引将对齐关系显式传给 logprob 处理器。

实现拆解

按 4 步拆解实现过程：

1. 放开调度器 admission：在 python/sglang/srt/managers/scheduler.py 的 handle_generate_request 中，删除原先对 DSpark + return_logprob 的三元硬拒绝分支（error_msg 直接设为拒绝文案），统一走 validate_dflash_request 校验其余 dflash 约束。这是让带 logprob 的请求能进入 DSpark 调度队列的入口前提。
2. 移除 worker 入口拒绝并新增索引缓存：在 python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py 的 forward_batch_generation 中删除 getattr(batch, "return_logprob", False) 时的 ValueError 抛出；新增 _linear_accept_indices 方法，按 bs * verify_num_draft_tokens 生成 (bs, verify_num_draft_tokens) 的 int64 线性索引，并用 self._linear_accept_index_cache 缓存 arange 结果，仅容量不足时重建，避免 decode 热路径反复分配。
3. 在 verify/accept 后接入 logprob 计算：同样在 dspark_worker_v2.py 的 _forward_decode 中，accept_and_finalize 返回后若 batch.return_logprob 为真，调用

compute_spec_v2_logprobs(batch, logits_output, accept.out_tokens.reshape(-1),self._linear_accept_indices(bs), self.verify_num_draft_tokens - 1)。折叠后的out_tokens 铺平成 1D，配合线性索引让 logprob 处理器把每个被接受 token 对齐到对应logits 行。该分支只在 return_logprob 请求上激活，普通请求无额外开销。

4. 恢复并扩充测试覆盖：在 test/registered/core/test_basic_sanity_dspark.py 中引入 sglang.test.kits.spec_server_kits 的 SpecLogprobKit mixin，删除被 @unittest.skip 禁用的 test_grammar_logprob_count_matches_completion_tokens（原为覆盖 SpecGrammarKit 的 logprob 校验，因 admission 拒绝被停用）。测试环境仍保持 SGLANG_RAGGED_VERIFY_MODE=compact，等于用 compact 折叠布局验证了 logprob 对齐的正确性。

关键文件：

- python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py（模块 推测解码；类别 source；类型 dependency-wiring；符号 _linear_accept_indices, forward_batch_generation, _forward_decode）：DSpark worker 核心：移除 forward_batch_generation 对 return_logprob 的硬拒绝，新增 _linear_accept_indices 索引缓存，在 _forward_decode 的 accept_and_finalize 之后接入 compute_spec_v2_logprobs，是 logprobs 能力实现的主载体。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 handle_generate_request）：handle_generate_request 删除 DSpark + return_logprob 的 admission 硬拦截，是带 logprob 请求能进入 DSpark 调度队列的入口前提。
- test/registered/core/test_basic_sanity_dspark.py（模块 回归测试；类别 test；类型 test-coverage；符号 TestBasicSanityDSpark, SpecLogprobKit）：引入 SpecLogprobKit 并移除被 skip 的 test_grammar_logprob_count_matches_completion_tokens，恢复 DSpark logprob 的注册式回归覆盖；测试环境保持 compact ragged-verify 模式，验证折叠布局下的 logprob 对齐。

关键符号：_linear_accept_indices, forward_batch_generation, _forward_decode, compute_spec_v2_logprobs, handle_generate_request

关键源码片段

python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py

DSpark worker 核心：移除 forward_batch_generation 对 return_logprob 的硬拒绝，新增 _linear_accept_indices 索引缓存，在 _forward_decode 的 accept_and_finalize 之后接入 compute_spec_v2_logprobs，是 logprobs 能力实现的主载体。

```
def _linear_accept_indices(self, bs: int) -> torch.Tensor:
    # 生成 (bs, verify_num_draft_tokens) 的线性索引，用于把折叠后的
    # out_tokens 铺平后与 logits 的布局对齐，为每个被接受的 token 找到
    # 对应的 logits 行。
    num_indices = bs * self.verify_num_draft_tokens

    # 缓存按需扩容：decode 热路径上每次都重新 torch.arange 会引入
```

```
    # 不必要的分配开销，因此仅在容量不足时重建索引缓存。
    if (
        self._linear_accept_index_cache is None
        or self._linear_accept_index_cache.numel() < num_indices
    ):
        self._linear_accept_index_cache = torch.arange(
            num_indices, dtype=torch.int64, device=self.device
        )
    return self._linear_accept_index_cache[:num_indices].view(
        bs, self.verify_num_draft_tokens
    )


def forward_batch_generation(
    self,
    batch: ScheduleBatch,
    on_publish=None,
    grammar_barrier=None,
) -> GenerationBatchResult:
    # 原来这里会对 return_logprob 直接抛 ValueError，现在已移除；
    # DSpark 的 decode 路径会在 verify 之后补算 logprob。
    if batch.forward_mode.is_extend() or batch.is_extend_in_batch:
        self._verify_planner.note_non_decode_step()
        self._observers.note_prefill_step()
        return self._forward_prefill(batch, on_publish)

    return self._forward_decode(batch, on_publish, grammar_barrier)

# 以下位于 _forward_decode 的 verify 完成之后（accept_and_finalize 返回后）：
# 仅当请求要求 logprobs 时才激活，避免给普通 decode 请求增加额外开销。
# out_tokens.reshape(-1) 与 _linear_accept_indices(bs) 的 1D 视图配合，
# 让 compute_spec_v2_logprobs 能把 accepted token 与对应 logits 行对齐，
# 写出 token_logprobs 与 top_logprobs。最后一个参数用于告知 logprob
# 处理器本次 verify 中每条序列的 logprob 窗口大小。
if batch.return_logprob:
    compute_spec_v2_logprobs(
        batch,
        logits_output,
        accept.out_tokens.reshape(-1),
        self._linear_accept_indices(bs),
        self.verify_num_draft_tokens - 1,
    )
```

python/sglang/srt/managers/scheduler.py

handle_generate_request 删除 DSpark + return_logprob 的 admission 硬拦截，是带 logprob 请求能进入 DSpark 调度队列的入口前提。

```
if self.spec_algorithm.is_dflash_family():
    # 之前对 DSpark + return_logprob 会走硬拒绝分支，现在 DSpark
```

```
# worker 已支持 logprob，统一交给 validate_dflash_request 校验
# 其它约束（如 overlap 相关限制），不再在这里做算法特判拦截。
error_msg = validate_dflash_request(req, self.enable_overlap)
if error_msg is not None:
    req.set_finish_with_abort(error_msg)
    self.init_req_max_new_tokens(req)
    self._add_request_to_queue(req)
    return
```

评论区精华

本 PR 没有产生代码 review 评论，设计权衡没有经过显式讨论，主要沉淀在代码与 PR body 中。Issue 线程里的沟通均为流程操作：QAQEthan 请求维护者添加 run-ci 标签触发 CI；hnyls2002 执行 /rerun-test 定向回归 test/registered/core/test_basic_sanity_dspark.py；github-actions[bot] 汇报 1-gpu-h100 runner 上通过。PR body 中列出了作者自测证据：短 / 长 completion 请求的 token/logprob 计数对齐、temperature=0 下的结构确定性与 determinism 检查、普通 DSpark 与 compact ragged-verify 两种模式均返回 HTTP 200 且包含 tokens、token_logprobs、top_logprobs。

- CI 触发与 DSpark sanity 回归验证 (other): 在 1-gpu-h100 runner 上重新运行 test_basic_sanity_dspark.py 通过，验证 DSpark logprob 支持没有破坏既有 sanity 覆盖。

风险与影响

- 风险：主要风险点如下：
  1. decode 热路径新增分支：_forward_decode 是每个 decode step 都会执行的路径，虽然 logprob 计算只在 batch.return_logprob 为真时激活，但一旦开启 logprob 的请求占比高，compute_spec_v2_logprobs 会持续叠加单步延迟。这是 logprob 能力的固有代价，但仍需关注 DSpark 高并发场景下的吞吐回归。
  2. 接受索引与 out_tokens 布局强耦合：_linear_accept_indices(bs) 的 (bs, verify_num_draft_tokens) 形状与 accept.out_tokens.reshape(-1) 的折叠布局必须严格匹配 accept_and_finalize 的输出约定。若未来调整 verify 布局（如新增 folded_commit 分支或 compact 回填逻辑），logprob 会静默错位而非报错。当前测试覆盖了 compact 模式，但未覆盖非 compact 的普通 ragged 模式。
  3. prefill/extend 路径 logprob 未显式覆盖：forward_batch_generation 移除拒绝后，extend 分支走 _forward_prefill，该路径没有调用 compute_spec_v2_logprobs，长文本请求在 extend 阶段的 logprob 完整性依赖 target_worker 自身行为，建议后续补充验证。
  4. admission 校验放宽：scheduler 不再对 DSpark return_logprob 做特判拦截，若 worker 某条边（如 idle/DP-attention 分支）遗漏 logprob 回填，会从 " 明确报错 " 退化为 " 静默返回不完整 logprobs"。 - 影响：用户侧：使用 DSpark 服务的调用方从 " return_logprob 直接被拒 " 变为可拿到 OpenAI 兼容的 tokens、token_logprobs、top_logprobs 字段，短 / 长 completion 请求均对齐。系统侧：调度器与 worker 各少一处特判分支，DSpark decode 路径多一个可选分支 + 一个按需扩容的索引缓存，非 logprob 请求几乎无感知。团队侧：DSpark 与依赖 logprob 的评测链路（如

SpecGrammarKit、GSM8K 评测）可组合使用，注册式 sanity 测试恢复了实质覆盖。整体影响范围限定在 speculative-decoding 的 DSpark 路径，不涉及其它算法与后端。 - 风险标记：decode 热路径新增分支，接受索引与 out_tokens 布局强耦合，prefill/extend 路径 logprob 未覆盖，admission 校验放宽

关联脉络

- PR #35057 [Spec] Point multi-layer eagle's last shared-read runner at the draft runner: 同属 speculative 模块的 worker 归属修正（multi_layer_eagle_worker_v2.py），与 dspark_worker_v2.py 共享 spec worker 的 verify 布局与 runner 生命周期语义，可对照理解 speculative 路径的演进。
- PR #34870 Fix swa eviction frontier for bigram keys: 涉及 speculative-decoding、scheduling、kv-cache 的缓存边界修复，与 DSpark/EAGLE 依赖的 radix cache 稳定性同属一条演进线，说明 speculative 系近期在 worker 归属、缓存边界、API 合同三个方向同时打磨。