PR #31753 完整报告

sgl-project/sglang

[DSPARK] Grammar-constrained decoding, incl. tool_choice=auto

合并时间：2026-07-25 20:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31753

执行摘要

- 一句话：DSPARK 推测解码支持语法约束与 tool_choice=auto
- 推荐动作：值得精读的 PR。它展示了如何在 speculative decoding 的 verify 阶段无缝集成 grammar mask，以及如何通过 step1（显式 grammar）和 step2（隐式 tool_choice）分步解决实际生产问题。特别关注 fold_eligible 与 CUDA graph 交互的设计决策，以及 supports_grammar_overlap 方法的多态扩展。

功能与动机

DSPARK speculative decoding 之前对携带 grammar 约束的请求（如 response_format 、 json_schema 、 regex 、 ebnf ）返回 HTTP 400 拒绝。实际上 DSPARK 的 target verify 路径已经具备应用 vocab mask 的能力，只是缺少 admission 守卫释放和 mask 应用。该 PR 是 #30096（DFLASH 语法约束）的配对补充，同时解决了生产环境中 tool_choice=auto 下标记泄露的问题。

实现拆解

1. 解除 admission 封锁（ python/sglang/srt/speculative/dflash_utils.py ）：validate_dflash_request 移除 spec_algorithm 参数和 spec_algorithm.supports_grammar_overlap() 的检查，不再拒绝 DSPARK 的 grammar 请求（DFLASH 仍拒绝）。同步更新调度器（ python/sglang/srt/managers/scheduler.py ）调用处，去掉 spec_algorithm 参数。
2. verify 阶段应用 vocab mask（ python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py ）：在 _forward_decode 中，当 batch.has_grammar 时，构建 GrammarTree.from_linear_chain （DSPARK 为线性验证树），调用 build_grammar_vocab_mask 生成掩码，并通过 draft_input.grammar.apply_vocab_mask 应用到 next_token_logits 。同时修改 fold_eligible 条件，排除 batch.has_grammar 的 batch，强制走 eager path 使得 mask 生效。
3. 扩展数据类（ python/sglang/srt/speculative/dflash_info_v2.py ）：DFlashDraftInputV2 新增 grammar 字段（类型 BaseGrammarObject），用于在 verify 过程中记录 grammar 对象。
4. 语法重叠能力声明（ python/sglang/srt/speculative/spec_info.py ）：supports_grammar_overlap 从 is_dflash() 改为 is_dflash_family()，使 DSPARK 与 DFLASH 一样上报支持 grammar 重叠。

5. 函数调用自动约束（第二个 commit，涉及 function_call_parser ）：should_constrain_auto 支持 DFLASH-family spec algorithm，通过 detector 暴露的 structural tag 触发 mask，解决 tool_choice=auto 下的标记泄露问题。
6. 测试配套（ test/registered/core/test_basic_sanity_dspark.py ）：添加 JSONConstrainedMixin 和 SpecGrammarKit 组合，以及一个被跳过的 test_grammar_logprob_count_matches_completion_tokens 因 admission 阶段拒绝 return_logprob 。同时存在 CPU 单元测试 test_dflash_validate_request.py 验证 admission 行为。

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 core-logic；符号 forward_batch_generation, _forward_decode）：核心实现文件：在 _forward_decode 中添加 grammar mask 逻辑，修改 fold_eligible 排除 grammar batch，新增 grammar_barrier 参数。是整个功能的关键。
- python/sglang/srt/speculative/dflash_utils.py（模块 推测解码；类别 source；类型 core-logic；符号 validate_dflash_request）：定义 admission 检验函数 validate_dflash_request，移除对 DSPARK 的 grammar 拒绝逻辑，使得 grammar 请求能被调度到 DSPARK worker。
- test/registered/core/test_basic_sanity_dspark.py（模块 集成测试；类别 test；类型 test-coverage；符号 test_grammar_logprob_count_matches_completion_tokens）：集成测试：通过 mixin 组合 JSONConstrainedMixin 和 SpecGrammarKit 确保 DSPARK 能够处理 grammar 请求，并新增被跳过的 test_grammar_logprob_count_matches_completion_tokens 作为已知限制的标记。

关键符号：validate_dflash_request, forward_batch_generation, _forward_decode, supports_grammar_overlap, should_constrain_auto

关键源码片段

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

核心实现文件：在 _forward_decode 中添加 grammar mask 逻辑，修改 fold_eligible 排除 grammar batch，新增 grammar_barrier 参数。是整个功能的关键。

```
# python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py
# 摘要：_forward_decode 中新增 grammar mask 逻辑

class DSparkWorkerV2(BaseSpecWorker):
    def _forward_decode(
        self, batch: ScheduleBatch, on_publish, grammar_barrier=None
    ) -> GenerationBatchResult:
        # ... 构建 verify_ids_2d, draft_input ...

        # 构建 grammar tree（DSPARK 是线性链，每个位置只有一个子节点）
        grammar_tree = (
            GrammarTree.from_linear_chain(verify_ids_2d) if batch.has_grammar else None
        )
```

```
        # fold_eligible 原条件；现在增加 not batch.has_grammar
        # 当有 grammar 时，fold 会跳过 CUDA graph epilogue，保证 mask 生效
        fold_eligible = (
            self._verify_executor.verify_epilogue is not None
            and proposal.folded
            and verify_logits_adjustments_are_noop(sampling_info)
            and self._simulate_acc_len <= 0
            and not batch.has_grammar # 新增：有 grammar 时强制 eager
        )

        with self._observers.segment(InfoSegment.TARGET_VERIFY):
            # ... 运行 verify launch ...
            logits_output = target_verify.logits_output

            if batch.has_grammar:
                # grammar barrier 非 None 时等待 host 侧 FSM 前进
                if grammar_barrier is not None:
                    grammar_barrier()
                # 构建 vocab mask，复用 EAGLE/NGRAM 路径的 build_grammar_vocab_mask
                vocab_mask = build_grammar_vocab_mask(
                    reqs=batch.reqs,
                    verify_input=draft_input,
                    tree=grammar_tree,
                    sampling_info=sampling_info,
                    device=logits_output.next_token_logits.device,
                )
                if vocab_mask is not None:
                    draft_input.grammar.apply_vocab_mask(
                        logits=logits_output.next_token_logits, vocab_mask=vocab_mask
                    )
```

python/sglang/srt/speculative/dflash_utils.py

定义 admission 检验函数 validate_dflash_request，移除对 DSPARK 的 grammar 拒绝逻辑，使得 grammar 请求能被调度到 DSPARK worker。

```
# python/sglang/srt/speculative/dflash_utils.py
# 变更后 validate_dflash_request 签名和逻辑

def validate_dflash_request(req: Req, enable_overlap: bool) -> Optional[str]:
    """
    检查 DFLASH-family 请求是否可接受。
    现在 DSPARK 的 grammar 请求不被拒绝，因为 DSPARK worker 支持 mask。
    """
    if req.return_logprob:
        return "DFLASH speculative decoding does not support return_logprob yet."

    if enable_overlap and req.return_hidden_states:
        return "DFLASH speculative decoding does not support return_hidden_states yet."
```

```
# grammar 拒绝检查已移除，因为 DSPARK 支持；DFLASH 仍然不支持，
# 但 DFLASH 的拒绝由其他机制处理（仍然在 admission 出错时返回错误）。
return None
```

评论区精华

1. tool_choice=auto 的补充（作者 shanemort1982）：第一个 commit 只处理了显式 grammar 请求，但 tool_choice=auto 不携带 grammar 对象，导致生产环境工具调用出现 5-8 倍泄露。第二个 commit 通过 should_constrain_auto 扩展解决了此问题，并附上 7 小时生产验证零泄露的数据。
2. CI 触发与测试通过（作者与合并者 hnyls2002 ）：合并者通过 /rerun-test 命令执行了特定测试（ test_basic_sanity_dspark.py 、 test_dflash_validate_request.py 、 test_dflash.py ），全部通过。
3. 重构与冲突解决（hnyls2002 的后续提交）：多次合并 main 分支解决冲突，尤其是 main 上 FunctionCallParser 已经重构为使用 get_auto_tool_call_structural_tag() ，需要适配。前作者（shanemort1982）的 Codex review 也指出需要跟踪最新 main 的变更。

- tool_choice=auto 的补充 (other): 合并者接受了此扩展，两个 commit 合并一起发布。
- CI 测试触发与通过 (testing): CI 测试通过，无新增失败。
- 与 main 分支的冲突解决 (other): 最终提交 7fd4eb2 中冲突解决，新逻辑适配了 main 的接口。

风险与影响

- 风险：
  1. 回归风险：对 DFLASH 的 grammar 拒绝行为无变化（仍拒绝），风险低。DSPARK 的非 grammar 请求路径完全跳过 mask，无性能损失。但 fold_eligible 新增条件 not batch.has_grammar 可能遗漏某些其他需要 eager 路径的场景，需观察。
  2. 安全 / 正确性：grammar mask 在 eager 路径应用，但如果 mask 构建失败（如 vocab_mask is None）则静默跳过，可能输出不合约束的内容。当前处理是 if vocab_mask is not None，没有 fallback 到拒绝响应。
  3. 性能：仅在有 grammar 的 batch 中增加一次 vocab mask 构建和应用，开销与 EAGLE 路径一致，但强制退出了 CUDA graph 折叠路径，可能略微降低吞吐。PR 作者评估 negligible。
  4. 兼容性：对非 spec 模式无影响。函数调用自动约束的 should_constrain_auto 扩展对非 DFLASH-family 算法无影响。 - 影响：用户：DSPARK 部署现在可以使用结构化输出和工具调用，完全解锁 grammar-constrained 场景。对已有 DSPARK 用户是重要功能补全。系统：引入的条件分支与 mask 计算仅影响 grammar 请求，非 grammar 流量零开销。团队：与 #30096 形成完整 DFLASH-family grammar 支持；新增的 grammar 字段和 should_constrain_auto 逻辑可能需要在后续 spec 算法中维护。测试新增的 CPU 单元测试是轻量的。
- 风险标记：核心路径变更，语法掩码集成，工具调用约束，CUDA graph 折叠条件修改

关联脉络

- PR #30096 [DFLASH] Support grammar-constrained decoding in speculative verify: 本 PR 是 #30096 的 DSPARK 配对实现，共享类似的验证时 mask 机制和 admission 调整。
- PR #30155 [xgrammar] vocab-mask on the multi-layer EAGLE verify path: 使用了相同的 generate_token_bitmask / apply_vocab_mask 设施。