PR #24999 完整报告

sgl-project/sglang

Add extension points on SpeculativeAlgorithm for custom spec v2

合并时间：2026-05-16 06:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24999

执行摘要

- 一句话：为自定义 speculative 算法提供扩展点
- 推荐动作：值得精读的设计模式改动。对于计划开发或集成自定义 speculative 算法的工程师，了解这两个扩展点是必须的。合并审批干净，CI 通过即可合入。建议后续为扩展点补充单元测试，确保新算法集成时不引入回归。

功能与动机

为支持自定义 speculative 算法 (custom spec v2)，需要对框架中现有多处硬编码的枚举判断 (如 is_dflash()) 和 FutureMap 直接构造进行抽象化，使得自定义算法可以通过重写 SpeculativeAlgorithm 的方法来定制行为，减少调用侧修改。

实现拆解

1. 在 spec_info.py 中新增两个扩展方法：supports_target_verify_for_draft() 默认返回 self.is_dflash()，供 draft worker 判断是否允许 TARGET_VERIFY 模式；create_future_map() 封装了 FutureMap 的构造逻辑（传入 self 作为 spec_algo 参数），默认实现与之前直接构造一致。
2. 调整 scheduler.py 中的 init_overlap 方法：将直接调用 FutureMap(...) 替换为 self.spec_algorithm.create_future_map(...)，不再直接 import FutureMap，使得 FutureMap 的构建可由算法子类控制。
3. 修改 cuda_graph_runner.py 中的 draft worker 检查：将 self.model_runner.spec_algorithm.is_dflash() 替换为 self.model_runner.spec_algorithm.supports_target_verify_for_draft()，从而允许自定义算法通过重写该方法决定是否支持 draft verify。
4. 修改 cpu_graph_runner.py 中的断言：将直接比较 == SpeculativeAlgorithm.NONE 替换为 is_none() 方法调用，消除对 SpeculativeAlgorithm 枚举类型的直接依赖。
5. 调整 overlap_utils.py 中 FutureMap 构造函数的签名：将 spec_algo 参数从 Optional[SpeculativeAlgorithm] = None 改为非可选类型 SpeculativeAlgorithm，因为调用方总是会传入有效算法。

关键文件：

- python/sglang/srt/speculative/spec_info.py（模块 推测解码；类别 source；类型 core-logic；符号 supports_target_verify_for_draft, create_future_map）：核心变更文件

，新增了两个扩展方法 supports_target_verify_for_draft 和 create_future_map，是抽象化的主要载体。

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 dependency-wiring）：调度器 init_overlap 中替换 FutureMap 直接构造为调用 spec_algorithm.create_future_map()，是集成扩展点的关键调用方。
- python/sglang/srt/model_executor/cuda_graph_runner.py（模块 CUDA Graph Runner；类别 source；类型 data-contract）：CUDA Graph Runner 中将硬编码的 is_dflash() 检查替换为 supports_target_verify_for_draft() 方法，使 draft worker 的 TARGET_VERIFY 判断可扩展。

关键符号：supports_target_verify_for_draft, create_future_map, init_overlap

关键源码片段

python/sglang/srt/speculative/spec_info.py

核心变更文件，新增了两个扩展方法 supports_target_verify_for_draft 和 create_future_map，是抽象化的主要载体。

```
# python/sglang/srt/speculative/spec_info.py
# 新增两个扩展方法，使自定义 speculative 算法可以通过重写来定制行为

def supports_target_verify_for_draft(self) -> bool:
    # 默认实现：只有 DFLASH 算法支持 draft worker 使用 TARGET_VERIFY 模式
    # 自定义算法可以复写此方法返回 True 以启用该模式
    return self.is_dflash()

def create_future_map(
    self,
    max_running_requests: int,
    chunked_prefill_size: int,
    context_len: int,
    device: torch.device,
) -> FutureMap:
    # 延迟导入避免循环依赖
    from sglang.srt.managers.overlap_utils import FutureMap
    # 创建 FutureMap 并传入 self 作为 spec_algo 参数
    return FutureMap(
        max_running_requests,
        chunked_prefill_size,
        context_len,
        device,
        self, # spec_algo 参数，现在为必选
    )
```

python/sglang/srt/managers/scheduler.py

调度器 init_overlap 中替换 FutureMap 直接构造为调用 spec_algorithm.create_future_map()，是集成扩展点的关键调用方。

```
# python/sglang/srt/managers/scheduler.py
# 在 init_overlap 方法中，通过多态调用创建 FutureMap

def init_overlap(self):
    # ... 省略部分代码 (MLX, stream 初始化 ) ...
    if not self.enable_overlap:
        self.future_map = None
        return

    # 之前 : self.future_map = FutureMap(self.max_running_requests, ...)
    # 现在通过算法对象创建，允许自定义算法复写
    self.future_map = self.spec_algorithm.create_future_map(
        self.max_running_requests,
        self.chunked_prefill_size,
        self.model_config.context_len,
        self.device,
    )
    self.batch_record_buf = [None] * 2
    self.batch_record_ct = 0
```

python/sglang/srt/model_executor/cuda_graph_runner.py

CUDA Graph Runner 中将硬编码的 is_dflash() 检查替换为 supports_target_verify_for_draft() 方法，使 draft worker 的 TARGET_VERIFY 判断可扩展。

```
# python/sglang/srt/model_executor/cuda_graph_runner.py
# 在 __init__ 方法中检查 draft worker 是否支持 TARGET_VERIFY 模式

if model_runner.spec_algorithm.is_speculative():
    if self.model_runner.is_draft_worker:
        # 之前 : if not self.model_runner.spec_algorithm.is_dflash():
        if (
            not self.model_runner.spec_algorithm.supports_target_verify_for_draft()
        ):
            raise RuntimeError("This should not happen")
    self.capture_forward_mode = ForwardMode.TARGET_VERIFY
    self.num_tokens_per_bs = self.speculative_num_draft_tokens
```

评论区精华

本条变更没有审核评论；合并者 merrymercy 直接批准，未提出讨论点。

- 暂无高价值评论线程

风险与影响

- 风险：本 PR 修改了 5 个文件，涉及调度器和推理图运行时的核心逻辑。风险包括：
  1. 行为一致性问题：create_future_map 和 supports_target_verify_for_draft 的默认实现与原有逻辑一致，但任何自定义算法复写后可能引发非预期行为。

2. 缺少测试覆盖：PR 未包含单元测试，对于扩展点的正确性和兼容性缺少验证；如自定义算法引入后出现回归难以被自动化捕获。
3. FutureMap 构造函数签名变更：从 Optional 变为必选，虽然当前所有调用方都提供有效参数，但若存在未发现的调用路径，会导致运行时错误。
4. 兼容性：此 PR 是纯重构，不改变外部行为，但下游用户若直接依赖 spec_algo 为 None 可能出现兼容问题。 - 影响：影响范围：涵盖 speculative decoding 的核心调度和推理图运行时。影响对象主要是未来开发自定义 speculative 算法的工程师；对现有用户无功能性影响。影响程度：中等。只影响开发扩展性，不改动现有逻辑行为。对系统影响：少量性能开销（多一次虚函数调用，可以忽略）。对团队影响：降低了增加新算法的工作量，提高了代码可维护性。

- 风险标记：缺少测试覆盖 , 核心路径变更

关联脉络

- PR #25125 [Disagg] Add retry with exponential backoff for prefill bootstrap register: 同属 speculative/scheduling 相关改进，但本 PR 是纯重构扩展点，与重试逻辑无直接关联。