PR #24055 完整报告

sgl-project/sglang

[SPEC][5/N] feat: batchsize-aware support for adaptive speculative_num_steps

合并时间：2026-06-06 06:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24055

执行摘要

- 一句话：为自适应推测解码增加 batch size 感知的独立 EMA 与步长切换
- 推荐动作：该 PR 扩展了自适应推测解码的核心能力，设计上考虑了向后兼容和性能优化（CUDA 图裁剪）。代码质量较高，测试覆盖充分。建议所有涉及推测解码的开发者精读，尤其是 AdaptiveStepSlot 和 AdaptiveController 的设计，以及 CUDA 图裁剪的实现思路。

功能与动机

PR #21599 introduced adaptive speculative decoding that dynamically adjusts speculative_num_steps based on EMA of accept lengths. This PR extends the adaptive controller to maintain per-batch-size independent state, where each BS range has its own EMA tracker, candidate steps, and hysteresis thresholds.

实现拆解

1. 配置结构改造（adaptive_spec_params.py）：将单一全局配置扩展为按 batch size 下限分组的字典。新增 _load_adaptive_config 函数加载和验证配置文件，支持新格式（{"1": {...}, "64": {...}}）和向后兼容的单条目格式。定义 DEFAULT_ADAPTIVE_CONFIG 涵盖三种典型场景。
2. 引入 AdaptiveStepSlot 和 AdaptiveSpeculativeParams：每个 BS 范围对应一个 AdaptiveStepSlot 实例，独立维护 EMA、候选步长和滞回参数。AdaptiveSpeculativeParams 管理所有 slot，并基于 bisect_right 查找当前 BS 对应的 slot。
3. 运行时状态切换（adaptive_runtime_state.py）：AdaptiveController 新增 activate_step_by_batch 方法，在每次 decode 前根据 batch size 选择最佳步数并切换运行时状态（attention backend、CUDA 图等）。on_verify_complete 方法现在接收 batch_size 参数，更新对应 slot 的 EMA。
4. Worker 集成（eagle_worker.py, eagle_worker_v2.py, base_spec_worker.py）：在 forward_batch_generation 的 decode 分支首行调用 activate_step_by_batch(batch_size)。build_adaptive_runtime_state 新增 cuda_graph_bs 参数，支持按步长裁剪 CUDA 图 BS 范围。_override_worker_state 临时修改 server_args，使 CUDA 图捕获只生成实际需要的 BS 图。

5. 测试与文档：新增 TestAdaptiveStepSlot 和 TestAdaptiveSpeculativeParams 单元测试，涵盖配置加载、步长校验、ceiling_coeff 等。更新 adaptive_speculative_decoding.mdx 文档，说明新配置格式和使用方法。

关键文件：

- python/sglang/srt/speculative/adaptive_spec_params.py（模块 参数配置；类别 source；类型 dependency-wiring；符号 load_adaptive_config, _load_adaptive_config, _resolve_candidate_steps, validate_adaptive_initial_steps）：配置文件格式改造、DEFAULT_ADAPTIVE_CONFIG、AdaptiveStepSlot 类定义，是 batch size 感知的核心参数管理模块。
- python/sglang/srt/speculative/adaptive_runtime_state.py（模块 运行时状态；类别 source；类型 core-logic；符号 init_states, on_verify_complete, activate_step_by_batch）：AdaptiveController 新增按 batch size 切换步数的核心逻辑，耦合参数管理与 Worker 运行时状态切换。
- test/registered/unit/spec/test_adaptive_spec_params.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestAdaptiveSpeculativeParams, _make_params_from_config, test_params_loads_config_path, TestAdaptiveStepSlot）：全面测试 AdaptiveStepSlot 和 AdaptiveSpeculativeParams 的配置加载、EMA 更新、ceiling_coeff 等。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测 Worker；类别 source；类型 core-logic；符号 on_verify_complete_cpu, activate_step_by_batch）：SpecV2 Worker 接入按 batch size 切换步数逻辑，修改 on_verify_complete_cpu 和新增 activate_step_by_batch 方法。
- python/sglang/srt/speculative/eagle_worker.py（模块 推测 Worker；类别 source；类型 core-logic）：SpecV1 Worker 对应修改，与 SpecV2 同步适配，支持 cuda_graph_bs 裁剪和状态切换。

关键符号：AdaptiveStepSlot.init, AdaptiveStepSlot.update, AdaptiveSpeculativeParams.init, AdaptiveSpeculativeParams.get_steps_for_batch, AdaptiveSpeculativeParams.on_verify_complete, AdaptiveController.activate_step_by_batch, AdaptiveController.on_verify_complete, EagleWorkerV2.activate_step_by_batch, EagleWorker.activate_step_by_batch, _load_adaptive_config, validate_adaptive_initial_steps

关键源码片段

python/sglang/srt/speculative/adaptive_spec_params.py

配置文件格式改造、DEFAULT_ADAPTIVE_CONFIG、AdaptiveStepSlot 类定义，是 batch size 感知的核心参数管理模块。

```
# -- 默认配置：按 batch size 下限分组，每组指定候选步长和滞回参数 --
# BS [1,8)：保守策略，步长可达 7，向下滞回 -0.25 抑制频繁降级
# BS [8,32)：中等策略，只取步长 1,3
# BS >=32：激进策略，固定步长 1（无推测，仅单步验证）
DEFAULT_ADAPTIVE_CONFIG: dict[str, dict] = {
    "1": {"candidate_steps": [1, 3, 7], "up_hysteresis": 0.0, "down_hysteresis": -0.25, "ceiling_
```

```
    coeff": 0},
    "8": {"candidate_steps": [1, 3] , "up_hysteresis": 0.0, "down_hysteresis": 0.0 , "ceiling_coeff":
    0},
    "32": {"candidate_steps": [1] , "up_hysteresis": 0.0, "down_hysteresis": 0.0 , "ceiling_coeff": 0}
    ,
}


def _load_adaptive_config(
    cfg_path: str | None,
) -> tuple[dict, dict[int, dict]]:
    """加载并验证自适应配置。
    当 cfg_path 为 None 时使用 DEFAULT_ADAPTIVE_CONFIG。
    返回 (原始配置 dict, 按 BS 下限解析的条目 dict)。
    """
    if cfg_path is not None:
        with open(cfg_path) as f:
            cfg = json.load(f)
    else:
        cfg = DEFAULT_ADAPTIVE_CONFIG

    bs_entries: dict[int, dict] = {}
    for key, entry in cfg.items():
        if not key.isdigit():
            continue # 跳过非数字键（如 ema_alpha 等全局键，不再支持）
        steps = entry.get("candidate_steps")
        # 校验：候选步长必须为正整数列表且非空
        if (
            not isinstance(steps, list)
            or not steps
            or not all(isinstance(s, int) and s > 0 for s in steps)
        ):
            raise ValueError(
                f"BS {key}: candidate_steps must be a list of positive ints, got {steps!r}"
            )
        bs_entries[int(key)] = entry

    if not bs_entries:
        raise ValueError(
            "speculative_adaptive_config must contain at least one integer-string "
            'BS key, e.g. {"1": {"candidate_steps": [1,3,7]}}. '
            f"Got keys: {list(cfg.keys())}"
        )
    return cfg, bs_entries
```

评论区精华

- CUDA 图裁剪：EanWang211123 指出当前逻辑为所有 BS 捕获所有步长的 CUDA 图，浪费显存。后续通过 init_states 中的 cuda_graph_bs 参数实现按步长裁剪 BS 范围。

- init_max_bs 与 FA3 一致性：Qiaolin-Yu 质疑不同 max_bs 初始化为何影响注意力结果。maodoudou168 解释 FA3 metadata 初始化依赖 max_bs，不同步长可能使用不同 pruned max_bs 导致相同 BS 下不同步数行为不一致。最终移除 init_max_bs，并计划在后续 PR 提供 anti-pingpong 机制。
- 初始步长校验：Qiaolin-Yu 询问为何不再将 initial_steps 自动加入候选列表。alphabetc1 解释因为有多个 BS slot，不清楚该加入哪个 slot，因此改为直接验证 initial_steps 必须在候选列表中，否则报错。
- activate_step_by_batch 集中化：alphabetc1 建议将步骤切换逻辑集中到一处。maodoudou168 将其封装到 activate_step_by_batch 方法中，在 decode 分支统一调用。
- CUDA 图捕获冗余：不同 BS 对应不同步长时，应为每个步长只捕获实际所需的 BS 范围 (performance): 已实现按步长裁剪：init_states 中传入 cuda_graph_bs，并在 _override_worker_state 中临时修改 server_args.cuda_graph_bs 以控制捕获范围。
- init_max_bs 导致不同步长的 FA3 attention 结果不一致 (correctness): 移除了 init_max_bs 参数，使所有步长使用统一的 max_bs；后续计划通过 anti-pingpong 机制解决剩余抖动。
- 初始步长校验：不再将 initial_steps 自动加入候选列表 (design): 添加 validate_adaptive_initial_steps 函数，若 initial_steps 不在任何 BS slot 的候选列表中则抛出 ValueError。
- 步骤切换逻辑集中化 (design): 封装为 activate_step_by_batch 方法，在 worker 的 decode 分支首行统一调用。

风险与影响

- 风险：
  - 兼容性风险：旧的全局配置格式（{"candidate_steps": ...}）与新的 per-BS 格式（{"1": {...}}）不兼容。用户升级时需要调整配置，但 PR 提供了向后兼容的单个 BS=1 条目模拟旧行为。
  - 性能风险：每轮 decode 调用 activate_step_by_batch 增加一次 dict 查找和整数比较，开销可忽略。
  - CUDA 图裁剪风险：若某个步长在 prune 后无任何 BS 使用，其 CUDA 图不会被捕获，但逻辑确保只在有效步骤间切换，不会引用未捕获的图。
  - 多 Worker 一致性：SpecV1 和 SpecV2 都需要修改，本 PR 提供了测试覆盖核心逻辑，但若测试覆盖不全可能导致回归。
- 影响：
  - 用户影响：使用 --speculative-adaptive 的用户将自动受益于 batch size 感知的优化，无需额外配置；也可通过 --speculative-adaptive-config 文件精细化控制。
  - 系统影响：每个 decode round 前增加一次步骤切换，开销极低；CUDA 图占用内存减少（因只捕获每个步长实际使用的 BS 范围）。
  - 团队影响：维护两套配置格式（旧全局 vs 新 per-BS）和向后兼容逻辑；后续可能需要统一配置路径。
  - 风险标记：配置格式兼容性，CUDA 图裁剪风险，多 Worker 一致性

关联脉络

- PR #21599 feat: adaptive speculative_num_steps: 本 PR 的前置工作，引入了全局自适应推测解码，本 PR 扩展为 per-BS 版本。
- PR #26354 Fix get_alloc_len_per_decode for adaptive spec: 修复自适应推测解码下 KV pool 分配偏移问题，本 PR 部分逻辑依赖该修复（review 中提及后续 rebase）。