PR #35861 完整报告

sgl-project/sglang

docs: add DSPARK speculative decoding option to Ling-3.0-flash cookbook

合并时间：2026-08-22 02:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35861

执行摘要

本 PR 为 Ling-3.0-flash cookbook 的 Deploy 面板新增 Spec Decode 一级选择器，把原先隐式的 NEXTN 投机解码升级为 NEXTN / DSPARK / Off 三档，并为每个 spec 提供独立基准行。核心实现是 ling-3.0-flash.jsx 中新增的 spec 匹配维度与 IIFE + dsparkTwin 自动生成 DSPARK 配方（固定 --linear-replayssm-cache-len 32），以及共享 Playground 组件对 ReplaySSM flag 的识别与清理。实测 B200 BF16 下 DSPARK 相对 NEXTN 吞吐提升约 2.1×（c1）与 1.35×（c16），GSM8K 精度基本持平。纯文档变更，无引擎代码改动。

功能与动机

PR body 的目标是把投机解码从隐性配置提升为显式选项："Promotes speculative decoding on the Ling-3.0-flash cookbook to a first-class Spec Decodeselector in the Deploy panel — NEXTN (built-in MTP) / DSPARK (external draft) / Off"。

动机有两点：

1. 性能收益：DSPARK 外部草稿路径在 B200 BF16 上比内置 MTP 的 NEXTN 快 2.1×（c1 吞吐 1902 vs 886 tok/s/GPU）与 1.35×（c16 8025 vs 5936），平均接受长度 6.22，而 GSM8K 精度几乎持平（96.59% vs 96.66%）。
2. 配方可运行性：草稿 block size 8 使验证窗口为 9 tokens，KDA ReplaySSM ring 必须是 2 的幂且不小于 2× 窗口，默认的 16 会直接触发启动校验失败——该参数是作者实际运行组合命令时发现并推导出来的。

实现拆解

1. 新增 spec 匹配维度（ling-3.0-flash.jsx）：引入 matchDims 中的 spec 维度（nextn / dspark / off），Deploy 面板据此渲染 Spec Decode 选择器；既有 low-latency 配方补 spec: "nextn"，high-throughput 补 spec: "off"（饱和场景下 draft/verify 开销不划算）。
2. IIFE + dsparkTwin 生成 DSPARK twin：Mintlify snippet 编译器只求值导出的表达式、丢弃顶层模块代码，因此 cells 整体迁入 IIFE。dsparkTwin 从每个 NEXTN low-latency cell 复制 twin，把 --speculative-algorithm NEXTN 替换为 DSPARK_FLAGS（DSPARK 算法 + 草稿路径 + ReplaySSM + cache-len 32），并只在 b200 + bf16 上标 verified: true。
3. 共享 Playground 对齐（_playground.jsx）：deriveFromBase 的识别列表与 stripFlagsByFirstToken 的剥离列表补上 --enable-linear-replayssm-spec 与 --linear-replayssm-cache-len，保证从 DSPARK 切回 NEXTN / Off 时无残留 flag；该组件被所有模型页面共享，改动不影响其他模型。

4. 基准按 spec 键化（ling-3.0-flash-benchmarks.jsx）：每条 match 补 spec 字段，与 cells 形成一致的数据契约；B200 BF16 low-latency 基于 PR #33561 head 0e5e40d8f（dev-Ling-3.0-flash 镜像）重新测量，新增 NEXTN / DSPARK 两套速度与精度数据。
5. 文档正文配套（Ling-3.0-flash.mdx）：Available Models 增补 DSPARK draft checkpoint；Configuration Tips 说明 ReplaySSM ring 参数推导（block size 8 → 9-token 验证窗口 → 2 的幂且 ≥ 18 → 32），并给出验证环境与数据。

测试与配套：无引擎测试变更；验证链路为 docs 自带 check_cookbook_configs.mjs、mint validate、mint broken-links，以及作者在 4×B200 上端到端 serving、smoke、bench 与全量 GSM8K 跑分。

docs/src/snippets/configs/inclusionAI/ling-3.0-flash.jsx

本 PR 的核心实现文件：新增 spec 匹配维度与 DSPARK 选项，cells 迁移到 IIFE 并用 dsparkTwin 自动生成 DSPARK twin 配方。

```
// Mintlify 的 snippet 编译器只求值导出的表达式，顶层模块代码会被整体丢弃，
// 因此 cells 必须放进 IIFE，flatMap 生成 DSPARK twin 的逻辑才能被执行。
cells: (() => {
  // DSPARK 外部草稿路径由 4 个 flag 联动：
  // --speculative-algorithm DSPARK 切到外部草稿算法；
  // --enable-linear-replayssm-spec 启用 KDA ReplaySSM 线性验证路径；
  // --linear-replayssm-cache-len 32 是推导值：草稿 block size 8 使 verify
  // 窗口为 9 tokens，ReplaySSM ring 必须是 2 的幂且 >= 2× 窗口，
  // 默认 16 过小，服务会在启动校验时拒绝。
  const DSPARK_FLAGS = [
    "--speculative-algorithm DSPARK",
    "--speculative-draft-model-path inclusionAI/Ling-3.0-flash-dspark",
    "--enable-linear-replayssm-spec",
    "--linear-replayssm-cache-len 32",
  ];

  // 从 NEXTN low-latency cell 复制出 DSPARK twin：
  // match 键改为 spec: "dspark"，flags 中出现的 NEXTN 整体替换为 DSPARK_FLAGS。
  const dsparkTwin = (cell, verified) => ({
    ...cell,
    verified,
    match: { ...cell.match, spec: "dspark" },
    flags: cell.flags.flatMap((f) =>
      f === "--speculative-algorithm NEXTN" ? DSPARK_FLAGS : [f]),
  });

  // 原有 low-latency 配方补上 spec: "nextn" 字段，flags 不变；
  // 下面仅保留 b200 + bf16 的完整 cell 作示例，其余硬件同构。
  const lowLatencyCells = [
    {
      match: { hw: "b200", variant: "default", quant: "bf16",
               strategy: "low-latency", spec: "nextn", nodes: "single" },
      verified: true,
```

```
        flags: [
          "--model-path {{MODEL_NAME}}",
          "--tp 4",
          "--speculative-algorithm NEXTN",
          "--mem-fraction-static 0.8",
          "--host {{HOST_IP}}",
          "--port {{PORT}}",
        ],
      },
      // ...h20-3e / h200 / h800 / h100 / gb300 以及 fp8 系列同构
    ];

    // 每个 low-latency cell 后紧跟其 DSPARK twin；
    // 只有 b200 + bf16 组合在 4×B200 上做过完整验证，标 verified。
    return [
      ...lowLatencyCells.flatMap((c) => [
        c,
        dsparkTwin(c, c.match.hw === "b200" && c.match.quant === "bf16"),
      ]),
      // ...high-throughput（spec: "off"）与 hicache（spec: "nextn"）系列 cell
    ];
  })(),
```

docs/src/snippets/_playground.jsx

共享的 Playground 组件：speculative 轴的 flag 识别与剥离列表新增 ReplaySSM 参数，保证切换 spec 时不留过期 flag，是所有模型页面共用的交互引擎。

```
  // ---- Axis: Speculative Decoding ---------------------------------------
  // 单选项轴：value 为 "current" 时保持 base 原样；为 "off" 时剥离全部
  // spec flag（greedy）；为具体选项时先剥离旧 flag，再插入选中项自己的 flags。
  // deriveFromBase：收集 cell 里现有的 --speculative-* flag 集合，
  // 与各选项的 flags 做数量 + 内容双重匹配，命中即视为该选项，否则回退 "current"。
  deriveFromBase: (cell, fc) => {
    const flags = (cell && cell.flags) || [];
    const baseSpec = flags.filter((f) => {
      const head = f.split(/[\s=]/)[0];
      return head === "--speculative-algorithm"
        || head === "--speculative-num-steps"
        || head === "--speculative-eagle-topk"
        || head === "--speculative-num-draft-tokens"
        || head === "--speculative-dspark-block-size"
        // ReplaySSM 是 DSPARK 外部草稿的配套验证路径，必须纳入识别与剥离，
        // 否则从 DSPARK 切回 NEXTN / Off 时会残留 --linear-replayssm-cache-len。
        || head === "--enable-linear-replayssm-spec"
        || head === "--linear-replayssm-cache-len"
        || head === "--speculative-ngram-max-bfs-breadth";
    });
    if (baseSpec.length === 0) return "off";
    for (const opt of (fc.options || [])) {
```

```
      if (!opt.flags || opt.flags.length !== baseSpec.length) continue;
      const ok = opt.flags.every((pf) => baseSpec.includes(pf));
      if (ok) return opt.id;
    }
    return "current";
  },

  // apply：切换选项时按首个 token 剥离全部 spec 相关 flag，
  // 再插入目标选项的 preset flags；选择与 base 一致时直接返回以保留 flag 位置。
  apply: ({ flags, env, value, fc, sel, h, derived }) => {
    if (value === "current") return { flags, env };
    if (derived && value === derived) return { flags, env };
    const picked = (fc.options || []).find((p) => p.id === value);
    if (picked && h.evaluateChip(picked, {
      ...sel,
      dpAttnOn: h.hasFlag(flags, "--enable-dp-attention"),
    }).disabled) {
      return { flags, env };
    }
    flags = h.stripFlagsByFirstToken(flags, [
      "--speculative-algorithm", "--speculative-num-steps",
      "--speculative-eagle-topk", "--speculative-num-draft-tokens",
      "--speculative-dspark-block-size", "--enable-linear-replayssm-spec",
      "--linear-replayssm-cache-len",
      "--speculative-ngram-max-bfs-breadth",
    ]);
    const preset = (fc.options || []).find((p) => p.id === value);
    if (preset?.flags?.length) flags = h.insertBeforeTail(flags, preset.flags);
    return { flags, env };
  },
```

评论区精华

本 PR 几乎没有人工 review 交锋：唯一的 review 是 zijiexia 的 APPROVED（空 body），唯一的 PR 评论来自 mintlify[bot] 的 preview 部署通知，不涉及技术内容。

真正有价值的信息藏在 8 个 commit 的演进里：

- commit 255df4a "fix Ling cookbook render — keep snippet config free of top-level code" 揭示了核心实现约束——Mintlify snippet 编译器只求值导出表达式，顶层模块代码会被丢弃，因此 cells 必须迁入 IIFE，dsparkTwin 也只能写在其中。
- commit ec31e57 "drop stray semicolon" 说明作者在渲染链路中还处理了残留分号问题；多次 retrigger mintlify preview 反映文档预览系统的迭代成本。

这些 commit 本身就是与构建约束的 " 讨论 "，也是本 PR 最有学习价值的部分。

风险与影响

风险：

1. 草稿仓库未公开：PR body 明确说明 draft repo 当前为私有，HF 链接匿名访问会 404。文档发布后用户按配方操作会先遇到 404，属于已知的临时性外部依赖风险，建议仓库公开后回访更新。
2. 文档内数据不一致：PR body 表格为 NEXTN 96.66% / DSPARK 96.59%，mdx 正文为 DSPARK 96.66% / NEXTN 96.44%（stop rate 99.77% / 99.62%），两处可能来自不同测量轮次，读者对照时容易困惑。
3. 共享组件匹配逻辑脆弱：deriveFromBase 要求 flag 数量与内容完全相等才判中；未来若引擎给 cell 追加其它 speculative flag，长度不匹配会让选择器回退为 "current"，表现为默认态异常。
4. 基准数据易过期：所有数据锚定 PR #33561 具体 commit 与 dev-Ling-3.0-flash 镜像，mdx 已留有 GB300 数据待补的 TODO，引擎演进后需要重新测量。

影响：用户侧是 Ling-3.0-flash 部署页新增 Spec Decode 维度，可一键切换并查看各自基准；系统侧无运行时影响；团队侧为其他模型 cookbook 扩展外部草稿投机解码提供了可复制的样板模式（matchDims 新维度 + IIFE 生成 twin）。

关联脉络

本 PR 是 Ling-3.0-flash cookbook 系列的延伸，直接依赖 PR #33561 的配方与基准（benchmarks 文件锚定其多个 commit，验证使用 dev-Ling-3.0-flash 镜像）。它与仓库中 speculative-decoding 功能的演进方向一致：从 NEXTN 内置 MTP 覆盖到 DSPARK 外部草稿 + KDA ReplaySSM 验证，把引擎能力沉淀为文档站的一等配置项。值得注意的是，改动将共享的 _playground.jsx 的 flag 生命周期管理又推进了一步，后续其他模型接入新的投机解码算法时，需要同步维护识别与剥离列表。