PR #33556 完整报告

sgl-project/sglang

Add Ling-3.0-flash cookbook

合并时间：2026-08-05 22:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33556

执行摘要

本 PR 为 inclusionAI/Ling-3.0-flash 混合 MoE 模型新增 SGLang Cookbook 页面，提供 6 类硬件上 BF16/FP8 的单节点部署配方，并重构了文档站点的 HiCache 交互组件以支持配置继承。核心配方已在 B200、GB300、H200 上通过 GSM8K 验证，部分硬件仍标记为未验证。

功能与动机

PR body 明确说明“Add the SGLang Cookbook page for inclusionAI/Ling-3.0-flash”。该模型是 124B 总参 /5.1B 激活的混合 KDA+MLA MoE，默认思考模式并支持工具调用，但此前缺少部署指引。本 PR 通过 Cookbook 页面和 Playground 交互组件，使用户能一键生成经过验证的启动命令。

实现拆解

1. 新增模型配置与基准数据：创建 ling-3.0-flash.jsx 和 ling-3.0-flash-benchmarks.jsx，定义硬件 / 量化 / 策略矩阵与逐 cell flags，并记录各 cell 的 GSM8K 结果。
2. 编写 Cookbook 页面：新增 Ling-3.0-flash.mdx，导入 Deployment/Playground 组件，包含模型介绍、配置提示、推理与工具调用示例、HiCache+Mooncake 部署说明。
3. 重构 Playground 的 HiCache 轴：_playground.jsx 引入 deriveFromBase 与 hasOverride，控件默认继承 base cell 状态，避免覆盖已验证配方；同时支持后端声明的 requiredFlags/requiredEnv 合并和 hasAutoBackend 继承选择。
4. 扩展部署组件与导航：_deployment.jsx 新增 H20-3e/H800 硬件目录和 dockerHostNetworkWhen 回调；docs.json 和 intro.mdx 更新导航与首页卡片。
5. 验证：无新增自动化测试，但通过 cookbook 配置校验和多种硬件的 GSM8K 实测验证，FP8 配方采用 TP+EP 以规避分片限制。

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

新增的模型配置，定义了全部硬件 / 量化 / 策略的部署 cells 和 Playground 功能，是 PR 的核心产物。

```
// Ling-3.0-flash Cookbook 配置（节选）
// 每个 cell 通过 match 元组唯一标识，flags 为实际启动参数，
// verified 表示是否经过完整 GSM8K 门禁验证。
export const config = {
  // 三种部署策略：低延迟（NEXTN 投机解码）、高吞吐（关闭投机）、HiCache+Mooncake
  strategies: [
```

```
  { id: "low-latency", label: "Low-Latency" },
  { id: "high-throughput", label: "High-Throughput" },
  { id: "hicache", label: "HiCache + Mooncake" },
],

// HiCache 卡片配置：声明默认后端与必选 flags/env，
// 组件将这些与选中 cell 的 flags 合并（见 _playground.jsx 的 hicache 轴）
playgroundFeatures: {
  hicache: {
    defaultBackend: "mooncake",
    requiredFlags: [
      "--mamba-scheduler-strategy extra_buffer", // 混合 KDA 调度必需
      "--enable-cache-report",
    ],
    backends: [
      {
        id: "mooncake",
        label: "Mooncake",
        flags: [
          "--hicache-storage-backend-extra-config '{\"hicache_storage_pass_prefix_keys\":true}'",
        ],
        env: [
          "MOONCAKE_MASTER={{MOONCAKE_MASTER}}",
          "MOONCAKE_PROTOCOL=tcp",
          "MC_MS_AUTO_DISC=0",
          "MOONCAKE_DEVICE=",
          "MOONCAKE_TE_META_DATA_SERVER={{MOONCAKE_METADATA_SERVER}}",
          "MOONCAKE_GLOBAL_SEGMENT_SIZE=0",
        ],
      },
    ],
  },
},

// 示例 cell：B200 + BF16 + Low-Latency（已验证）
cells: [
  {
    match: { hw: "b200", variant: "default", quant: "bf16", strategy: "low-latency", nodes:
    "single" },
    verified: true,
    env: ["SGLANG_ALLOW_OVERWRITE_LONGER_CONTEXT_LEN=1"],
    flags: [
      "--model-path {{MODEL_NAME}}",
      "--tp 4", // 141GB 级 GPU 用 TP4
      "--context-length 262144", // YaRN 扩展到 256K
      "--speculative-algorithm NEXTN", // 内置 MTP 投机解码
      "--json-model-override-args '{\"rope_scaling\":{\"rope_type\":\"yarn\",\"factor\":2.0,...}}'",
      "--mem-fraction-static 0.8", // 为 CUDA graphs 留出余量
      "--host {{HOST_IP}}",
```

```
        "--port {{PORT}}",
      ],
    },
    // 其余 cell 遵循相同结构，仅在 hw/quant/strategy 与 flags 上变化
  ],
};
```

docs/src/snippets/_playground.jsx

修改了 HiCache 轴状态继承逻辑，是本次改动中唯一的交互逻辑变更，影响所有使用 Playground 的页面。

```
// HiCache 轴的核心逻辑（重构后）
// 通过 deriveFromBase 提取 base cell 中的 HiCache 状态，
// apply 仅在用户显式覆盖时才重写 flags/env，避免破坏已验证配方。
hicache: {
  initState: () => ({ enable: null, backend: null, writePolicy: "auto" }),

  deriveFromBase: (cell, fc, h) => {
    const flags = (cell && cell.flags) || [];
    return {
      enable: h.hasFlag(flags, "--enable-hierarchical-cache"),
      backend: h.findFlagArg(flags, "--hicache-storage-backend"),
      writePolicy: h.findFlagArg(flags, "--hicache-write-policy") || "auto",
    };
  },

  apply: ({ flags, env, value, fc, sel, h, derived }) => {
    if (fc.excludesHw && sel && fc.excludesHw.includes(sel.hw)) return { flags, env };
    // Deploy 面板托管启用开关时（showWhen），base 已携带完整配方，
    // 这里只修改暴露的两个旋钮，避免静默更改 ratio/layout/io-backend。
    if (typeof fc.showWhen === "function") {
      const set = (name, val) => {
        flags = h.stripFlagsByFirstToken(flags, [name]);
        if (val) flags = h.insertBeforeTail(flags, [`${name} ${val}`]);
      };
      if (value.backend) set("--hicache-storage-backend", value.backend);
      if (value.writePolicy && value.writePolicy !== "auto") {
        set("--hicache-write-policy", value.writePolicy);
      }
      return { flags, env };
    }

    // 关键重构：只有当用户改动任一开关时才进入重写分支，
    // 否则原样返回 base（保证未触碰的配方字节级不变）。
    const hasOverride = value.enable !== null
      || value.backend !== null
      || (value.writePolicy && value.writePolicy !== "auto");
    if (!hasOverride) return { flags, env };
```

```
    // 收集所有需要剥离的 flag/env 前缀，包括后端自定义的 requiredXXX
    const backendOptions = fc.backends || [];
    const ownedHeads = [
      "--enable-hierarchical-cache", "--hicache-ratio", "--hicache-size",
      "--hicache-write-policy", "--hicache-mem-layout", "--hicache-io-backend",
      "--hicache-storage-backend", "--hicache-storage-prefetch-policy",
      "--hicache-storage-backend-extra-config",
      ...((fc.requiredFlags || []).map((f) => f.split(/\s/)[0])),
      ...backendOptions.flatMap((o) => (o.flags || []).map((f) => f.split(/\s/)[0])),
    ];
    const ownedEnvKeys = [
      ...(fc.requiredEnv || []),
      ...backendOptions.flatMap((o) => o.env || []),
    ].map((e) => e.split("=")[0]);
    flags = h.stripFlagsByFirstToken(flags, ownedHeads);
    if (ownedEnvKeys.length) env = h.stripEnvByPrefix(env, ownedEnvKeys);

    // 状态计算：显式值优先，否则继承 base 或配置默认
    const enabled = value.enable !== null
      ? value.enable : !!(derived && derived.enable);
    const backend = value.backend !== null
      ? value.backend
      : ((derived && derived.backend) || fc.defaultBackend || null);

    if (enabled) {
      // AMD 特判与 flags 组装（与 base 相同，此处省略具体分支）
      // ... 最终通过 h.insertBeforeTail 追加生成的 flags/env
    }
    return { flags, env };
  },

  // render 中新增 hasAutoBackend，决定后端选择是否可继承：
  // 若存在 id:null 的 auto 选项，则默认显示继承而非 derived 值
  render: ({ axisId, value, setValue, fc, base, s, renderChip, renderSelect, derived }) => {
    // ...
    const hasAutoBackend = (fc.backends || []).some((o) => o.id === null);
    const backend = value.backend !== null
      ? value.backend
      : (hasAutoBackend ? null : ((derived && derived.backend) || fc.defaultBackend || null));
    // ...
  },
},
```

评论区精华

本 PR 没有 review 评论，唯一的 APPROVED 来自 zijiexia（空 body）。关键决策体现在 commit 信息：FP8 配方从 TP2 调整为 TP+EP 以避免 blockwise E4M3 分片限制；移除 --default-chat-template-kwargs 工作区；HiCache 轴重构为继承 base 以避免覆盖已验证配方。

风险与影响

- 风险：_playground.jsx 是共享组件，HiCache 轴重构影响所有使用 Playground 的页面；dockerHostNetworkWhen 若配置错误可能选用 host 网络；benchmarks 中 GB300 条目数值缺失，H20-3e/H800/H100 未验证。
- 影响：用户获得开箱即用的部署指引；文档系统获得更灵活的配置继承能力；项目扩展 InclusionAI 模型矩阵。

关联脉络

本 PR 的运行验证基于 PR #33561，并吸收 #33712 的 HiCache cell 支持；与 Ring-2.6-1T 等 InclusionAI 系列 cookbook 组成完整的模型部署矩阵。