PR #23947 完整报告

sgl-project/sglang

[Docs] add cookbook for Ling-2.6 family

合并时间：2026-04-29 00:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23947

执行摘要

该 PR 为 SGLang 文档新增了 InclusionAI Ling-2.6 模型家族的 cookbook，包含两个交互式部署选择器组件（Ling-2.6-flash和Ling-2.6-1T）、一个主文档页面（MDX）和导航配置更新。已合并，但 review 自动化检查提出了 React import 缺失、变量未使用等未解决问题。

功能与动机

Ling-2.6 是 InclusionAI 的新一代语言模型，包含 104B/7.4B 活跃参数的 BF16 MoE 模型（flash）和约 1T 参数的 FP8 MoE 模型（1T）。PR body 指出两者共享 1:7 MLA + Lightning Linear hybrid attention backbone，旨在提升推理效率和 agent 工作负载表现。该 cookbook 提供部署命令生成器、thinking 模式说明和工具调用配置，降低用户使用门槛。

实现拆解

1. 新增 Ling-2.6-flash 部署选择器：ling-26-flash-deployment.jsx 创建 React 组件，提供硬件（H20-3e/H100/H200/B200）、YaRN 上下文长度（128K/256K）、工具调用 parser 和推理 parser 选项，根据选择生成 glang serve 命令。
2. 新增 Ling-2.6-1T 部署选择器：ling-26-1t-deployment.jsx 创建 React 组件，支持单节点（GB300/GB200 TP=4）和双节点（H200/B200 TP=8+PP=2）部署，包含相同的可选 parser。
3. 撰写 cookbook 文档：Ling-2.6.mdx 包含模型架构描述、性能数据、安装指引、部署命令（通过导入组件嵌入）、thinking 模式详细说明（默认关闭，需通过系统消息启用）和参考基准（GSM8K 96.21%）。
4. 更新导航配置：在 docs.json 的 InclusionAI 分组中添加 Ling-2.6 页面入口。
5. 修订与润色：经过 7 次后续提交，移除了 dead envPrefix 逻辑、统一使用 sglang serve CLI、修正 thinking 模式锚点 slug、调整安装章节措辞。

docs_new/src/snippets/autoregressive/ling-26-1t-deployment.jsx

核心组件，实现 1T 模型的双节点 / 单节点部署命令生成，包含工具调用和推理 parser 逻辑。

docs_new/src/snippets/autoregressive/ling-26-flash-deployment.jsx

flash 模型部署选择器，支持 H20/H100/H200/B200 单节点，并包含 YaRN 上下文长度切换。

关键源码片段

docs_new/src/snippets/autoregressive/ling-26-1t-deployment.jsx

核心组件，实现 1T 模型的双节点 / 单节点部署命令生成，包含工具调用和推理 parser 逻辑。

```
// Ling-2.6-1T 部署命令生成
const generateCommand = () => {
  const { hardware, toolcall, reasoning } = values;
  const isSingleNode = hardware === 'gb300' || hardware === 'gb200';

  // 辅助函数：追加模型加载优化与可选 parser（注意参数内嵌双引号需转义）
  const tail = (cmd) => {
    let out = cmd;
    out += ` \
  --model-loader-extra-config '{"enable_multithread_load":"true","num_threads":64}'`;
    if (toolcall === 'enabled') out += ` \
  --tool-call-parser qwen`;
    if (reasoning === 'enabled') out += ` \
  --reasoning-parser qwen3`;
    return out;
  };

  if (isSingleNode) {
    // 单节点：GB300 或 GB200，TP=4
    let cmd = `sglang serve \\n`;
    cmd += `  --model-path inclusionAI/Ling-2.6-1T \\n`;
    cmd += `  --tp-size 4 \\n`;
    cmd += `  --trust-remote-code \\n`;
    cmd += `  --host 0.0.0.0 \\n`;
    cmd += `  --port \${PORT}`;
    return tail(cmd);
  }

  // 双节点：H200 或 B200，TP=8 + PP=2
  const generateNodeCmd = (rank) => {
    let cmd = `sglang serve \\n`;
    cmd += `  --model-path inclusionAI/Ling-2.6-1T \\n`;
    cmd += `  --tp-size 8 \\n`;
    cmd += `  --pp-size 2 \\n`;
    cmd += `  --nnodes 2 \\n`;
    cmd += `  --node-rank ${rank} \\n`;
    cmd += `  --trust-remote-code \\n`;
    if (rank === 0) {
      cmd += `  --host 0.0.0.0 \\n`;
      cmd += `  --port \${PORT} \\n`;
    }
    cmd += `  --dist-init-addr \${MASTER_IP}:\${DIST_PORT}`;
    return tail(cmd);
  };

  let output = `# MASTER_IP 为节点 0 的 IP。PORT 和 DIST_PORT 可自行指定。

`;
```

```
  output += `# 节点 0:\n`;
  output += generateNodeCmd(0);
  output += `

# 节点 1:\n`;
  output += generateNodeCmd(1);
  return output;
};
```

docs_new/src/snippets/autoregressive/ling-26-flash-deployment.jsx

flash 模型部署选择器，支持 H20/H100/H200/B200 单节点，并包含 YaRN 上下文长度切换。

```
// Ling-2.6-flash 部署命令生成（始终单节点 TP=4）
const generateCommand = () => {
  const { yarn, toolcall, reasoning } = values;
  // 注意：hardware 选择未影响命令，所有硬件使用相同参数（可能未来扩展性能提示）
  let cmd = `sglang serve \\n`;
  cmd += `  --model-path inclusionAI/Ling-2.6-flash \\n`;
  cmd += `  --tp-size 4 \\n`;
  cmd += `  --trust-remote-code \\n`;
  cmd += `  --host 0.0.0.0 \\n`;
  cmd += `  --port \${PORT}`;
  if (yarn === 'enabled') {
    // YaRN 扩展到 256K 上下文
    cmd += ` \
  --context-length 262144`;
    cmd += ` \
  --json-model-override-args '{"rope_scaling": {"rope_type": "yarn", "factor": 2.0, "rope_theta":
  6000000, "partial_rotary_factor": 0.5, "original_max_position_embeddings": 131072}}'`;
  }
  if (toolcall === 'enabled') {
    cmd += ` \
  --tool-call-parser qwen25`;
  }
  if (reasoning === 'enabled') {
    cmd += ` \
  --reasoning-parser qwen3`;
  }
  return cmd;
};
```

评论区精华

> gemini-code-assist[bot]: “[Both components] use useState and useEffect hooks but does not import them from react. This will cause a ReferenceError at runtime unless these are provided globally by the build environment.” — 两个组件均缺少 React hooks import，构成构建风险。

> gemini-code-assist[bot]: "The logic for envPrefix is currently ineffective because isGB and isSingleNode are derived from the same hardware IDs. As a result, isGB && !isSingleNode is always false." — 发现 dead 代码，已在后续 commit 中清理。
>
> gemini-code-assist[bot]: "The hardware selection is extracted from values but never used to modify the generated command. If the command is indeed identical for all listed hardware, consider removing the selector." — 建议移除或有效利用 hardware 选择器，目前未处理。

风险与影响

- 构建风险：两个 JSX 组件缺少 React hooks import，虽可能被文档构建工具全局注入，但不符合标准要求，存在白屏风险。
- 命令准确性：flash 模型部署命令未进行实际验证（PR body 标注），用户可能遇到参数无效。同时 --model-loader-extra-config 的 JSON 转义方式在某些 shell 可能失败。
- 用户体验：flash 选择器中的 hardware 选项对命令无影响，用户切选项后命令不变可能造成困惑。
- 影响范围：仅文档站点变化，不影响 SGLang 运行时。对用户是正向引导，降低新模型部署门槛。

关联脉络

该 PR 与近期多个模型 cookbook（如 #23907 Nemotron 3 Nano Omni、#23945 MiMo V2.5 MTP）共同表明 SGLang 团队在扩展文档对新模型家族的支持，并逐渐采用交互式 JSX 部署选择器。这些 PR 通常遵循相同的模式：创建 JSX 片段、MDX 文档和更新 docs.json。此模式有助于降低模型集成文档的维护成本。