

PR #38839 完整报告

sgl-project/sglang

Fix the DeepSeek-V4.1 reasoning example and make every NVIDIA cell start

合并时间: 2026-09-10 16:49

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38839>

执行摘要

- 一句话: 修复 V4.1 Cookbook 推理示例并补 NVIDIA 启动内存配置
- 推荐动作: 值得精读, 但重点不在代码: 本 PR 的价值集中在 PR body 对 reasoning 参数契约的分析与 OOM 排查数据上, 任何要在 DeepSeek-V4.1 上接 reasoning 或 tool calling 的工程师都应读完 §3.1 相关段落, 只看 diff 会误以为它只是加了一个 flag。三个值得关注的设计决策: (1) 打开 thinking 走请求侧 reasoning_effort, 而不是要求用户设置服务端环境变量, 因为 normalize_reasoning_inputs 会自动同步 chat_template_kwargs.thinking; (2) OOM 修复首选收敛 --cuda-graph-max-bs-decode, 只在 H200 上叠加降低 --mem-fraction-static; (3) 文档里显式区分“能启动”与“已调优”, 避免读者把 64 当性能参数。建议动作: 复核合并后的 deepseek-v4_1.jsx, 确认 H200 / B200 cell 是否确实带上了 cap 与 verified, 必要时补一个后续 PR; 并考虑把真机验证矩阵以更机器可读的形式落盘, 减少人工表述与文件内容之间的漂移。

功能与动机

PR body 明确写出两个“把页面按原样跑到真机上”才发现的问题。其一是 §3.1 从不展示推理: “Thinking is off by default (SGLANG_DEFAULT_THINKING = EnvBool(False))”, serving_chat.py 只从 chat_template_kwargs.thinking 读这个开关, 关闭时 encoding_dsv41 用 收尾 assistant 轮次, 模型不产出 thinking 块, reasoning parser 也就拆不出内容——而示例既没传 reasoning_effort 也没传 chat_template_kwargs.thinking, 正好演示了“本节主题不出现”的情形。其二是启动失败: “Five of the eight NVIDIA cells did not start”, 每个 DSpark cell 加两个 H200 cell 都报 Exception: Capture cuda graph failed: CUDA out of memory., 派生的 --cuda-graph-max-bs-decode 在捕获解码图时放不下, 服务从未 ready, 其中还包含页面早已标记 verified 的 GB300 Low-Latency cell。此外正文两处表述不准确也被一并纠正: 请求侧不能传整数 budget (类型是 Optional[Union[ReasoningEffortTier, float in [0.0, 0.99]]], 整数会被 400 拒绝), 以及 none 不只是“warn 后回退”, 它还会关闭 thinking。

实现拆解

本次改动只涉及文档与 Playground 配置片段, 没有运行时源码与测试变更, 按以下 5 步落地。

1. 修正 §3.1 推理示例的 thinking 假设 (docs/cookbook/autoregressive/DeepSeek/DeepSeek-V4_1.mdx)

- 在 Reasoning Example 的 `client.chat.completions.create` 调用里补上 `reasoning_effort="high"`。
- 正文新增一句：parser 只能拆出模型真实产出的 thinking 块，thinking 默认关闭（`SGLANG_DEFAULT_THINKING=false`），任何非 none 的 `reasoning_effort` 会顺带把 `chat_template_kwargs.thinking` 打开。
- 影响：示例从“必然打印空 `reasoning_content`”变成与本节主题一致，读者不再需要猜测推理通道为何为空。

2. 重写 reasoning effort 参数契约段落（同一 mdx）

- 把“tier 或 integer budget”改为：请求侧只接受 low / high / xhigh / max 四个 tier，或 `[0.0, 0.99]` 的 float；整数会被 400 拒绝（`literal_error`，实测 `reasoning_effort=50`）。
- 补充整数 1–100 的合法路径是 `chat_template_kwargs.reasoning_effort`，但该字段在参数校验之后才被 pop 再赋值，本身不会打开 thinking，需与 `chat_template_kwargs.thinking` 成对使用。
- 把 none 从“warn 后回退”一组中拆出单独说明：它除了不是有效 tier，还会关闭 thinking——这正是读者最需要知道的一点。

3. 补齐 §3.2 与 §2 的文档配套

- §3.2 工具调用示例同样打印 `reasoning_content`（`finish_reason: "tool_calls"` 之外还检查推理通道），因此也加上 `reasoning_effort="high"`。
- §2 新增“Cap the decode CUDA graphs”提示，面向手写启动命令的读者：DSpark 解码图按派生值捕获会 OOM，需要显式收敛 `--cuda-graph-max-bs-decode`。

4. Playground 配置片段加 cap 并升级验证状态（docs/src/snippets/configs/deepseek-ai/deepseek-v4_1.jsx）

- B300 Low-Latency 补 `--cuda-graph-max-bs-decode 64`，并把 `verificationStatus: "in-progress"` 改成 `verified: true`；B300 High-Throughput 同样升级为 `verified: true`。
- 注释块按 B200 / B300 拆开，点名 B300 需要额外内存开关（派生值 256 在 0.8 显存占比下捕获 DSpark 解码图会 OOM），MI350X cell 完全不动。
- 需要复核的一处：PR body 声称 GB300 / H200 的 cell 也加了同样的 cap（H200 另加 `--mem-fraction-static 0.8`，因为仅加 cap 只到 1.89 GiB 可用对 2.00 GiB 分配），并把“All eight NVIDIA cells move to verified”；但本次提供的 diff 材料只覆盖 B200/B300 区域，其中 B300 两个 cell 的状态变更可见，H200 / B200 cell 仍写着 in-progress 且没有 cap，commit message 也只说“mark the B300 cells verified”。

5. 验证与配套（无测试代码）

- 没有新增测试；唯一自动检查是 `node docs/scripts/check_cookbook_configs.mjs`，作者报告结果为 `cookbook config check: OK`，它只校验配置结构，无法发现“命令合法但启动 OOM”。
- 真机验证矩阵：GB300 / B200 / H200 使用页面自带镜像 `lmsysorg/sglang:dev-dsv41`，B300 使用 #38798 的代码树（1aa0e96）通过 `PYTHONPATH` 叠加在 stock image 上；每一行“starts, serves”都用真实 completion 确认，而不只是 `/health`，因此这套验证只能人工重跑，进不了 CI。

关键文件：

- docs/src/snippets/configs/deepseek-ai/deepseek-v4_1.jsx（模块 示例配置；类别 config；类型 configuration；符号 b300-low-latency, b300-high-throughput, b200-low-latency, b200-high-throughput）：本 PR 中唯一会改变实际启动行为的文件：决定 NVIDIA 各 cell 能否挺过 DSpark 解码图捕获，并承载 verified 状态标记。
- docs/cookbook/autoregressive/DeepSeek/DeepSeek-V4_1.mdx（模块 使用文档；类别 docs；类型 documentation；符号 §3.1 Reasoning Example, §3.2 Tool Calling, §2 Cap the decode CUDA graphs）：用户直接复制的推理 / 工具调用示例与 reasoning_effort 契约说明都在这里，是“示例为什么打印空 reasoning_content”这一问题的修复主体。

关键符号：ChatCompletionRequest.normalize_reasoning_inputs, encoding_dsv41.encode_messages, SGLANG_DEFAULT_THINKING, REASONING_EFFORT_MAPPINGS

关键源码片段

docs/src/snippets/configs/deepseek-ai/deepseek-v4_1.jsx

本 PR 中唯一会改变实际启动行为的文件：决定 NVIDIA 各 cell 能否挺过 DSpark 解码图捕获，并承载 verified 状态标记。

```
// docs/src/snippets/configs/deepseek-ai/deepseek-v4_1.jsx
// B300 Low-Latency: 本次 diff 中唯一同时补上显存开关并升级验证状态的 cell。
// 64 是启动可行值而非调优值：它只保证解码图能捕获成功，不代表吞吐已调优。
{
  match: { hw: "b300", strategy: "low-latency" },
  nnodes: 1,
  verified: true, // 由 verificationStatus: 'in-progress' 升级为 verified: 真机跑通并返回真实 completion
  flags: [
    "--trust-remote-code",
    "--model-path {{MODEL_NAME}}",
    "--tp 4",
    "--ep-size 4",
    "--mem-fraction-static 0.8",
    "--speculative-algorithm DSPARK", // DSpark 会额外捕获解码图，是启动期 OOM 的直接来源
    "--speculative-dspark-block-size 5",
    // GB300 不需要、B300 需要：派生值 256 在该显存占比下捕获 DSpark 解码图会 OOM
    "--cuda-graph-max-bs-decode 64",
    "--reasoning-parser auto",
    "--tool-call-parser auto",
    "--host {{HOST_IP}}",
    "--port {{PORT}}",
  ],
},
```

// 注意：PR 描述还声称 H200 两个 cell 也加了同样的 cap 与 --mem-fraction-static 0.8、

```
// 并把全部八个 NVIDIA cell 升级为 verified, 但本次 diff 材料里看不到这两处改动,  
// 复核时请以合并后的文件内容为准。
```

docs/cookbook/autoregressive/DeepSeek/DeepSeek-V4_1.mdx

用户直接复制的推理 / 工具调用示例与 reasoning_effort 契约说明都在这里, 是“示例为什么打印空 reasoning_content”这一问题的修复主体。

```
# docs/cookbook/autoregressive/DeepSeek/DeepSeek-V4_1.mdx — §3.1 Reasoning  
Example (本 PR 修正后)  
from openai import OpenAI  
  
client = OpenAI(base_url="http://localhost:30000/v1", api_key="EMPTY")  
  
resp = client.chat.completions.create(  
    model="deepseek-ai/DeepSeek-V4.1-Flash",  
    messages=[{"role": "user", "content": "What is 15% of 240?"}],  
    # 关键修复: thinking 默认关闭 (SGLANG_DEFAULT_THINKING = EnvBool(False)) 。  
    # 不传该参数时, encoding_dsv41 用 </think> 收尾 assistant 轮次, 模型不产出 thinking 块,  
    # reasoning_parser 也就拆不出内容, reasoning_content 恒为空 —— 恰是本节要展示的东西。  
    # 任何非 'none' 的取值都会被 ChatCompletionRequest.normalize_reasoning_inputs 推导成  
    # chat_template_kwargs.thinking = True, 这是打开 thinking 的推荐路径。  
    reasoning_effort="high",  
)  
msg = resp.choices[0].message  
print("Reasoning:", getattr(msg, "reasoning_content", None)) # 修正前为空, 现在能看到思考文本  
print("Answer:", msg.content) # 实测输出: "15% of 240 is **36**."  
  
# 契约要点 (正文已同步改写):  
# - 请求侧只接受 low / high / xhigh / max, 或 [0.0, 0.99] 的 float;  
# - 整数 budget (1-100) 只能走 chat_template_kwargs.reasoning_effort,  
# 且该校验后回填的路径不会自动打开 thinking, 需与 chat_template_kwargs.thinking 配对;  
# - 'none' 会关闭 thinking, 而不是像 minimal / medium 那样仅 warn 后后退。
```

评论区精华

评审几乎是空白的: 唯一一条 review 来自 wisclmy0611, 状态为 APPROVED、body 为空, 没有任何行内评论; 合入者 zijixia 与作者是同一人, 等于没有第三方对 PR body 里的实测数字与改动范围做交叉验证。真正有信息量的是 PR body 中的三条自述式技术论证: 其一, 修复示例选 reasoning_effort 而不是 chat_template_kwargs.thinking, 因为 ChatCompletionRequest.normalize_reasoning_inputs 会从 effort != "none" 推导 thinking 并 setdefault 到 chat_template_kwargs, 这才是该模型打开 thinking 的惯用路径; 其二, 整数 budget 必须从文档里拆出去, 请求侧实测 reasoning_effort=50 返回 HTTP 400 literal_error, 整数 1-100 只能走 chat_template_kwargs.reasoning_effort, 而这条路径在校验之后被 pop 再赋值, 不会自动开 thinking; 其三, OOM 修复优先用 batch cap, 只有 H200 需要叠加降低 --mem-fraction-static, 作者也自陈 64 是 “startup-feasibility value, not a tuned one”。

- 唯一评审为无正文 APPROVED，未对实测数据做交叉验证 (other): 无争议直接合入，但 PR body 的真机数据与 diff 可见范围之间的差异没有被任何评审者指出。
- 打开推理走请求侧 reasoning_effort，而不是 chat_template_kwargs.thinking (design): 示例改用 reasoning_effort="high"，并在正文点名 thinking 默认关闭这一事实。
- 整数 reasoning budget 会被 400 拒绝，原文档表述有误 (correctness): 正文改写为列出合法 tier 与 float 区间，单独说明整数路径与 none 关闭 thinking 的语义。
- 启动 OOM 用 batch cap 还是继续压低显存占比 (performance): 以 --cuda-graph-max-bs-decode 64 为主，H200 叠加降低 --mem-fraction-static；参数未做性能调优。

风险与影响

- 风险：1) 文档描述与 diff 可见范围不一致，这是最需要复核的一点：PR body 称“All eight NVIDIA cells move to verified”且 H200 两个 cell 也加了 --cuda-graph-max-bs-decode 64 与 --mem-fraction-static 0.8，但 commit message 只说“mark the B300 cells verified”，提供的 diff 也只覆盖 B200/B300 注释拆分、B300 Low-Latency 的 cap 与两个 B300 cell 的 verified 变更。若 H200 / B200 实际未改，页面在这两类硬件上仍会启动失败，正是本 PR 要消灭的故障模式。2) 配置值被当作推荐值传播：作者明确说 64 只是启动可行值，但它在页面上是一个普通 flag，用户容易在 batch 超过 64 的场景里把它当性能配置使用（三个 High-Throughput cell 保持不设 cap）。3) 结论强绑定硬件与镜像：verified 的前提是 lmsysorg/sglang:dev-dsv41 与 #38798 的 1aa0e96，镜像或内核默认值变化后这些配方可能重新 OOM，而没有任何机制会提醒。4) 缺少自动回归：check_cookbook_configs.mjs 只能校验配置结构，对“命令能过检查但启动即 OOM”无能为力，本 PR 修的正是这一类问题。5) 契约描述依赖实现细节：文档对“整数 budget 不自动开 thinking”的解释建立在 chat_template_kwargs.reasoning_effort 在校验之后被 pop 再回填这一顺序上，上游若调整解析顺序，措辞会失效。6) 无性能、安全或数值风险——未修改任何运行时路径。
- 影响：用户侧影响为高：NVIDIA 侧 5 个按文档复制即失败的 cell 变为可用，§3.1 / §3.2 示例从“必然为空”变为真正能展示推理通道，对第一次部署 DeepSeek-V4.1 的团队直接决定首次跑通时间。系统侧影响为零：只有文档与 Playground 配置片段，可安全回滚，不涉及内核、调度或显存分配逻辑。团队侧影响中等：把 Cookbook 的 verified 标记从“看起来对”推进到“真机跑过”，同时把维护成本显性化——需要按平台 × 镜像 × commit 维度定期复验，而本 PR body 与 diff 的范围差异说明这套标记的准确性目前仍依赖作者的人工纪律。
- 风险标记：文档与 diff 范围可能不一致，配置值非调优值，缺少自动回归覆盖，验证结论强绑定镜像与硬件

关联脉络

- PR #38802 Add DeepSeek-V4.1 Flash cookbook: 本 PR 修正的正是 #38802 新增的两个文件（同一页面、同一 Playground 配置），动机段落明确以该页面“按原样执行”为前提。
- PR #38844 [Cookbook] DeepSeek-V4.1: add the HiCache L2 knob to the Playground: 触及同一文件 docs/src/snippets/configs/deepseek-ai/deepseek-v4_1.jsx，说明该 Playground 配置处于高频迭代，容易再次出现文档与真机不一致。

- PR #38798 (PR body 引用的 B300 验证基线代码树) : PR body 明确说明 B300 cell 的验证是在“#38798 tree (1aa0e96)”上通过 PYTHONPATH 叠加完成的, 是本 PR verified 结论的依赖之一; 该 PR 的标题未在材料中给出, 需以仓库实际内容为准。
- PR #38677 [AMD] Update v4 args for agentic workload: 同属 DeepSeek-V4 cookbook 的启动参数调优工作线 (deepseek-v4.jsx), 可对照其参数取舍与验证方式。