PR #33131 完整报告

sgl-project/sglang

feat(cookbook): add DGX Spark support for Inkling-Small

合并时间：2026-08-01 08:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33131

PR 分析报告：Inkling-Small 新增 DGX Spark 部署支持

执行摘要

本 PR 为 Inkling-Small 的部署 cookbook 增加 NVIDIA DGX Spark（GB10 / SM121）支持：两台 Spark 通过 ConnectX-7 互联、TP=2 运行 NVFP4 checkpoint，并配套专用 arm64 CUDA 13 镜像与 verified 启动配方。变更以文档和配置为主，同时改动了文档站点共用的部署命令生成引擎 _deployment.jsx 与交互面板 _playground.jsx，引入 multiNodeDockerFlags 机制，把平台级 docker 标志从提示注释提升为引擎注入的命令片段。影响范围限于文档站点，不触及推理运行时；CI extra 通道标记失败，具体原因未在材料中说明。

功能与动机

PR body 明确说明：Two Sparks over ConnectX-7 can run the NVFP4 checkpoint with TP=2, but the cookbook lacked a shared hardware catalog entry and a launch recipe for this platform。DGX Spark 是桌面级 Blackwell 平台，之前硬件目录没有对应条目，用户无法生成可复制的部署命令。作者声明 Accuracy/Speed 数据 N/A（docs/cookbook only），基准卡片以 stub 占位，等实测后补充。

实现拆解

1. 共享硬件目录扩展（docs_new/src/snippets/_deployment.jsx）：在 HARDWARE_CATALOG.blackwell 中新增 dgx-spark 条目（128 GB 统一内存），并把硬件条目 schema 扩展出 multiNodeDockerFlags 字段；新增 fabricFlagsOf(hwId) 解析函数，优先级为模型私有 config.hardware 优于共享目录，未声明时返回空数组。docker 模式下，多节点命令会在 --network host 之后注入该平台的 fabric 标志（--ulimit memlock=-1:-1、--cap-add IPC_LOCK、--device /dev/infiniband），单节点不注入。这是本 PR 的核心机制改动，让平台级 docker 标志成为硬件目录的固有属性。
2. Inkling-Small 模型配置（docs_new/src/snippets/configs/thinkingmachines/inkling-small.jsx）：supportedHardware 增加 dgx-spark；dockerImages 映射专用镜像 lmsysorg/sglang:dev-inkling-small-dgx-spark（arm64 CUDA 13、NCCL 2.30.7）；新增verifiedcell（hw:dgx-spark×strategy:balanced×quant:nvfp4×nodes:multi-2），flags 组合为 --tp 2、--attention-backend triton、--fp4-gemm-backend marlin、--moe-runner-backend marlin、--disable-prefill-cuda-graph 等；env 开启 SGLANG_ENABLE_UNIFIED_RADIX_TREE=1。同时更新 Playground gates：TP 旋钮新

增值 2（限 DGX Spark）、FlashInfer TRT-LLM / AITER / Triton MoE 后端对 dgx-spark 隐藏（SM121 无 FP4 runner，回退 Marlin W4A16）、PD 拆分对 dgx-spark 隐藏。

3. Playground 引擎同步（docs_new/src/snippets/_playground.jsx）：由于 Mintlify 会剥离模块级状态，Deployment 与 Playground 两个引擎无法共享同一常量表，因此在 renderCommandLines 内镜像 HW_MULTINODE_DOCKER_FLAGS 映射，多节点 docker 命令输出与 Deployment 引擎保持一致，注释明确要求两边同步修改。
4. 配套数据与文档：inkling-small-benchmarks.jsx 增加 dgx-spark × multi-2 占位条目（sglang_version: dev-inkling-small-dgx-spark，精度 TBD）；Inkling-Small.mdx 补充镜像拉取列表与部署要点；.claude/skills 下 cookbook-add-model、cookbook-review-pr、cookbook-migrate-model 四个技能文档同步更新 multiNodeDockerFlags 的字段语义、硬件目录表与 review 检查规则。
5. 测试配套：无新增自动化测试，PR 声明 docs/cookbook only；PR body 中 CI extra 通道标记失败，具体原因未说明。

docs_new/src/snippets/_deployment.jsx

共享部署命令生成引擎，新增 dgx-spark 硬件目录条目与 multiNodeDockerFlags 机制，fabricFlagsOf 在多节点 docker 命令中注入 ConnectX-7 RDMA 标志，是所有模型 cookbook 页共用的核心引擎。

```
// ---- 1. 硬件目录：DGX Spark 条目 ----
// GB10 Grace Blackwell，128 GB 统一内存（非独立 VRAM）。
// multiNodeDockerFlags 是该平台多节点互联所需的 docker run 标志：
// ConnectX-7 RDMA 需要 pinned memory（--ulimit memlock）与 IB 设备透传。
// 放在硬件条目上是因为它平台不变，任何模型选择 dgx-spark 都会自动携带。
const HARDWARE_CATALOG = {
  blackwell: [
    { id: 'b300', label: 'B300', vram: '288GB' },
    { id: 'gb300', label: 'GB300', vram: '288GB' },
    { id: 'b200', label: 'B200', vram: '192GB' },
    { id: 'gb200', label: 'GB200', vram: '192GB' },
    {
      id: 'dgx-spark', label: 'DGX Spark', vram: '128GB',
      multiNodeDockerFlags: [
        '--ulimit memlock=-1:-1', '--cap-add IPC_LOCK', '--device /dev/infiniband',
      ],
    },
  ],
  hopper: [/* … */],
  amd: [/* … */],
};

// ---- 2. fabric 标志解析 ----
// config.hardware（模型私有）优先于共享 HARDWARE_CATALOG；
// 两者都没有则返回空数组，保证老硬件零影响。
const fabricFlagsOf = (hwId) => {
```

```
  const extra = (config.hardware || []).find((h) => h.id === hwId);
  if (extra) return extra.multiNodeDockerFlags || [];
  for (const list of Object.values(HARDWARE_CATALOG)) {
    const hit = list.find((h) => h.id === hwId);
    if (hit) return hit.multiNodeDockerFlags || [];
  }
  return [];
};

// ---- 3. docker 命令组装（docker 模式） ----
const dockerLines = [
  ...gpuAccessLines,
  // 多节点需要 host 网络打通跨节点 rendezvous 端口与 NCCL/GLOO 流量；
  // 单节点只映射 serve 端口。
  multinode ? '  --network host' : `  -p ${servePort}:${servePort}`,
  // 仅多节点场景注入 fabric 标志，单节点不出现无意义的 --device /dev/infiniband。
  ...(multinode ? fabricFlagsOf(sel.hw).map((f) => '  ' + f) : []),
  '  -v ~/.cache/huggingface:/root/.cache/huggingface',
  ...(config.placeholders && config.placeholders.HF_TOKEN
    ? ['  --env "HF_TOKEN={{HF_TOKEN}}"'] : []),
  ...cellEnv.map((e) => '  --env ' + e),
  '  --ipc=host',
  `  ${image}`,
  '  sglang serve',
  ...flags.map((f) => '    ' + f),
];
```

docs_new/src/snippets/configs/thinkingmachines/inkling-small.jsx

Inkling-Small 部署配置主体：新增 DGX Spark 平台、专用镜像与 verified cell，并调整 Playground 各卡片的硬件 gate（TP/MoE/PD-disagg），是用户实际看到的部署配方来源。

```
// DGX Spark（GB10 / SM121）+ NVFP4 —— 2× Spark 通过 ConnectX-7 互联。
// 每节点 1 张 GPU，所以 TP=2 跨 2 节点；该平台无 FP4 专用 runner，
// MoE 回退 Marlin W4A16，attention 用 Triton 后端，并禁用 prefill CUDA graph。
{
  match: { hw: 'dgx-spark', variant: 'default', quant: 'nvfp4', strategy: 'balanced', nodes: 'multi-2' },
  verified: true,
  env: [
    'SGLANG_ENABLE_UNIFIED_RADIX_TREE=1',
  ],
  flags: [
    '--trust-remote-code',
    '--model-path {{MODEL_NAME}}',
    '--tp 2', // 2 节点 × 1 GPU
    '--quantization modelopt_fp4',
    '--attention-backend triton', // SM121 无 TRT-LLM FP4 路径
    '--page-size 128',
    '--fp4-gemm-backend marlin',
```

```
    '--moe-runner-backend marlin',
    '--mamba-radix-cache-strategy extra_buffer',
    '--mem-fraction-static 0.85', // 统一内存 128 GB，预留余量
    '--swa-full-tokens-ratio 0.1', // SWA 与 Mamba/sconv 池占比
    '--mamba-full-memory-ratio 0.1',
    '--enable-multimodal', // 兼容图片 / 音频输入
    '--disable-prefill-cuda-graph', // 该平台禁用 prefill CUDA graph
    '--reasoning-parser inkling',
    '--tool-call-parser inkling',
    '--host {{HOST_IP}}',
    '--port {{PORT}}',
  ],
},
```

docs_new/src/snippets/_playground.jsx

交互式部署面板的渲染引擎，与 _deployment.jsx 行为需保持一致；因 Mintlify 剥离模块状态无法共享常量，镜像了一份 DGX Spark fabric 标志表，是双引擎同步风险的所在。

```
// Mintlify 会剥掉模块级状态，Deployment 与 Playground 两个引擎
// 无法共享 _deployment.jsx 里的 HARDWARE_CATALOG，因此这里镜像一份
// 多节点 fabric 标志表；修改引擎时必须两边同步。
const HW_MULTINODE_DOCKER_FLAGS = {
  'dgx-spark': [
    '--ulimit memlock=-1:-1', '--cap-add IPC_LOCK', '--device /dev/infiniband',
  ],
};
const fabricFlags = HW_MULTINODE_DOCKER_FLAGS[sel.hw] || [];

const dockerLines = [
  'docker run --gpus all',
  '  --shm-size 32g',
  (multinode || pdMode) ? '  --network host' : `  -p ${servePort}:${servePort}`,
  // 多节点 docker 命令注入 fabric 标志，与 Deployment 引擎输出保持一致。
  ...(multinode ? fabricFlags.map((x) => '  ' + x) : []),
  '  -v ~/.cache/huggingface:/root/.cache/huggingface',
  '  --env "HF_TOKEN={{HF_TOKEN}}"',
  ...cellEnv.map((e) => '  --env ' + e),
  '  --ipc=host',
  `  ${image}`,
  '  sglang serve',
  ...f.map((x) => '    ' + x),
];
```

评论区精华

本 PR 没有公开的 review 评论（仅有的 issue 评论是 gemini-code-assist 的停服通知），但第二个提交承载了最有价值的设计决策。合并者 zijiexia 在 commit 4bca725 中说明：

> “The Deploy and Playground engines emit them into multi-node docker run commands, so DGX Spark's ConnectX-7 RDMA flags land in the command itself

> instead of the config's multiNodeHints, which rendered as # comments above both run modes — including the Python output, where docker flags don't belong."

这是一次"职责划分"的纠偏：平台不变的 docker 标志归属硬件条目 multiNodeDockerFlags，由引擎注入命令本身；面向用户的提示语才放 multiNodeHints。相关的 authoring 规范、review 规则与迁移文档都同步了这条边界。

风险与影响

- 共享引擎全局影响：_deployment.jsx 的 HARDWARE_CATALOG 与 fabricFlagsOf 对所有模型 cookbook 页面生效。得益于未声明 multiNodeDockerFlags 的硬件返回空数组，向后兼容，但属于共享代码路径变更，需回归验证其他模型的 docker 命令输出。
- 双引擎重复实现：Deployment 与 Playground 各自维护一份 fabric 标志表（一处读硬件目录、一处写死映射），未来新增需 fabric 标志的平台时若只改一边，两处命令输出会不一致，属于持续维护风险。
- 镜像可用性：专用镜像 lmsysorg/sglang:dev-inkling-small-dgx-spark 是否已发布、NCCL 2.30.7 组合是否验证过，材料未证实；cookbook 页面若推给用户而镜像缺失会造成部署失败。
- 基准数据缺失：benchmarks 仅 stub，无精度数据，用户无法评估该平台上的质量表现。
- CI extra 未通过：PR body 显示 extra run 失败，原因未说明，合入前应确认与本次改动无关。

关联脉络

- #33083（DeepSeek-V4 Flash cookbook 配方）：同期 cookbook 配方演进，改动同一批文档站点引擎文件（_playground.jsx、snippets/configs），本 PR 复用并扩展了该工作流。
- #33023（Inkling short convs 迁移到注意力后端）：Inkling 系列模型的注意力 / 卷积后端演进，本 PR 的 DGX Spark cell 使用 Triton attention 与 Mamba/sconv 池参数，与该架构的运行时能力直接相关。
- 整体方向：SGLang cookbook 正在从数据中心 GPU 场景扩展覆盖工作站 / 桌面级 Blackwell 平台（GB10），硬件目录与引擎机制逐步泛化，multiNodeDockerFlags 为后续新平台（如 RTX PRO 类）的多节点互联支持铺平了路。