

PR #37995 完整报告

sgl-project/sglang

docs(cookbook): Qwen3.8-Flash-Next NVFP4 recipes for DGX Spark (1x, 2x) and RTX PRO 6000

合并时间: 2026-09-08 06:33

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/37995>

执行摘要

本 PR 为 Qwen3.8-Flash-Next cookbook 页面扩展了 NVFP4 部署方案，新增了对 DGX Spark (GB10) 和 RTX PRO 6000 (RTX 6000) 两款新硬件的完整支持。变更涉及三个文档 / 配置文件，通过复杂的覆盖层逻辑和多轮硬件验证，为两类新硬件提供了包含准确度与性能基准的部署食谱，解决了大规模 NVFP4 检查点在统一内存和显存受限环境下的适配难题。

功能与动机

原始 cookbook 页面仅覆盖数据中心 GPU (H200, B200, B300, GB300, AMD MI350X/MI355X)。为支持新硬件部署，需要解决以下问题：

- DGX Spark (GB10, 128 GB 统一内存)：126 GiB 的 NVFP4 检查点 (78 GiB 权重 + 47.7 GiB FP8 N-gram 表) 无法单节点运行。方案是采用 TP=2 跨两台设备，或在单节点上使用文件后端 PLE 表 (--ple-offload-backend file, 参见 #37068)。
- RTX PRO 6000 (96 GB 离散显存)：必须将 47.7 GiB 的 N-gram 表卸载到主机固定内存 (--ple-offload-embedding) 才能容纳剩余权重。
- NVIDIA ModelOpt NVFP4 导出：需要新的加载器 (#38121) 和特定的 MoE 运行器后端标志。
- GB10 上的 MTP 崩溃：在低延迟 MTP 单元中发现输出退化为重复！，根因是 Triton 路由器内核在 PDL 等待前加载了偏置 (#36811, #38290)，需要在验证中使用已修复的镜像。

实现拆解

1. 硬件与量化配置扩展(docs/src/snippets/configs/Qwen/qwen3.8-flash-next.jsx): - 将 supportedHardware 扩展为包含 dgx-spark 和 rtx6000。 - 新增 nvfp4-nvda 量化选项，并添加 hardware 数组以定义 RTX PRO 6000 的本地属性。 - 引入 nodesOptions 中的 multi-2 选项，以支持多节点部署单元。
2. PLE Offload 覆盖层逻辑重写(docs/src/snippets/configs/Qwen/qwen3.8-flash-next.jsx):
 - 根据硬件类型和节点数，动态禁用或强制 auto、on、off 和新的 file 选项。例如，在 DGX Spark 上禁用 auto 和 on (统一内存无收益)，并根据节点数强制 off (多节点) 或 file (单节点)。
3. 添加验证单元与基准数据(docs/src/snippets/configs/Qwen/qwen3.8-flash-next-benchmarks.jsx):
 - 为 12 个新单元 (8 个 DGX Spark + 4 个 RTX PRO 6000) 添加了完整的基准测试数据，包括经过完整集验证的 GSM8K 准确度 (97.0%+) 和不同并发下的 TTFT、TPOT、吞

吐量指标。

4. 用户文档与注释完善(docs/cookbook/autoregressive/Qwen/Qwen3.8-Flash-Next.mdx):

- 更新安装说明，为 DGX Spark 和 RTX PRO 6000 指定专用的 dev-qwen38-next-local 镜像。
- 新增详尽的硬件特定手风琴部分，涵盖启动顺序、NCCL 版本、内存限制、并发控制、PLE 文件后端注意事项和 NVIDIA 导出差异。

docs/src/snippets/configs/Qwen/qwen3.8-flash-next.jsx

核心配置文件，包含了所有新增的硬件、量化、覆盖层逻辑和验证单元定义，是实现 PR 目标的关键。

```
// PLE Offload 覆盖层配置，展示了针对不同硬件的禁用逻辑和强制选项。
// 对于 DGX Spark: 统一内存使 RAM 卸载无收益，因此根据节点数强制 Off 或 "On (NVMe file)"。
// 对于 RTX PRO 6000: 96GB 显存必须卸载 47.7 GiB 的 N-gram 表，因此强制 On。
{
  id: "pleOffload",
  title: "PLE Offload",
  showWhen: (sel) => ![ "mi350x", "mi355x" ].includes(sel.hw),
  default: "auto",
  options: [
    {
      id: "auto", label: "Auto",
      // 在 DGX Spark 和 RTX PRO 6000 上禁用 Auto
      disabled: (sel) => sel.hw === "dgx-spark" || sel.hw === "rtx6000",
      disableReason: (sel) => sel.hw === "rtx6000"
        ? "RTX PRO 6000 (96 GB) 只能通过将 47.7 GiB FP8 N-gram 表放入固定主机 RAM 来运行此检查点..."
        : "DGX Spark 是统一内存: RAM 卸载无收益...",
      hints: ["PLE Offload: auto-enabled for BF16 on CUDA, off otherwise"]
    },
    {
      id: "on", label: "On",
      // 在 DGX Spark 上禁用 On
      disabled: (sel) => sel.hw === "dgx-spark",
      disableReason: "DGX Spark 是统一内存: RAM 卸载无收益...",
      flags: ["--ple-offload-embedding"]
    },
    {
      id: "off", label: "Off",
      // 在 RTX PRO 6000 或单节点 DGX Spark 上禁用 Off
      disabled: (sel) => sel.hw === "rtx6000" || (sel.hw === "dgx-spark" && sel.nodes === "single")
        ,
      disableReason: (sel) => sel.hw === "dgx-spark"
        ? "单个 DGX Spark 无法将 126 GiB 检查点放入其 128 GB 统一内存..."
        : "RTX PRO 6000 (96 GB) 无法同时容纳 47.7 GiB N-gram 表...",
      flags: ["--no-ple-offload-embedding"]
    },
  ],
  // 文件后端选项（仅对 DGX Spark 单节点可用）
}
```

```

{
  id: "file", label: "On (NVMe file)",
  showWhen: (sel) => sel.hw === "dgx-spark",
  disabled: (sel) => sel.nodes !== "single",
  disableReason: "文件后端表仅对单 Spark 单元进行了验证...",
  flags: ["--ple-offload-embedding", "--ple-offload-backend file"],
  hints: [
    "PLE 表 -> $SGLANG_CACHE_DIR/ple/<model> 下的稀疏 47.7 GiB 文件...",
    "启动前删除旧的表文件，直到上游修复为止：重写已填充文件速度慢..."
  ]
}
]
}
]
}

```

评论区精华

审查者 [zijiexia](#) 的审查意见驱动了多项重要修正：

- 准确性数据完整性：“gsm8k result across dgx and rtx6000 are measured on partial of the benchmark set, it is not a good reference for accuracy and not good for reproduction, maybe consider remove them or rerun the full set.” 作者因此重新运行了所有单元的完整 1,319 题集测试。
- 内存数值一致性：“the rtx 6000 resident-weight figure is stated three ways... If the ~80 / ~81 numbers are read off nvidia-smi after load and therefore include the CUDA context and allocator overhead rather than weights alone, that's worth saying once — otherwise the three should be unified.” 作者统一了文档中的数值表述。

风险与影响

- 配置复杂性风险：引入的硬件特定覆盖层逻辑（基于节点数的条件强制）增加了配置的维护成本和认知负荷。
- 文档同步风险：详细的硬件特定说明（如启动顺序、内存限制）依赖于特定的硬件 / 驱动版本，未来版本变化可能导致文档过时。
- 外部依赖：食谱依赖于未发布的开发分支（qwen4-main-squashed）和特定镜像（dev-qwen38-next-local），增加了外部依赖性。
- 用户影响：为两类新硬件提供了关键、经过验证的部署指南，显著降低了用户在这些平台上的部署门槛。

关联脉络

本 PR 是一个更广泛功能演进的一部分：

1. 内核修复：依赖 #36811 和 #38290 修复了导致 DGX Spark MTP 崩溃的路由器内核问题（#37111）。
2. 新功能启用：依赖 #37068（文件后端 PLE 表）和 #38121（NVIDIA ModelOpt 加载器）为新硬件启用了必要功能。

3. 文档扩展：延续了 #38374 (Qwen3.5 FP8 Blackwell) 的模式，为更多硬件和量化组合扩展了 cookbook 页面。

建议

这是一个高质量、验证严谨的文档 PR。其价值不仅在于提供了部署食谱，更在于其处理复杂硬件差异的配置逻辑、多轮验证的透明记录，以及解决实际部署问题（如 MTP 崩溃、PLE 文件后端）的详细说明。建议负责维护部署文档或支持新硬件平台的工程师精读此 PR，特别是其覆盖层配置逻辑和硬件特定注释部分，可作为未来类似工作的优秀范本。