PR #24515 完整报告

sgl-project/sglang

LPLB: linear-programming load balancer for MoE expert parallelism

合并时间：2026-06-17 01:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24515

执行摘要

- 一句话：新增 LPLB 线性规划负载均衡，优化 MoE 专家路由
- 推荐动作：这是一个高质量的 PR，核心 IPM 内核设计、空 rank 处理、性能优化（融合核、预分配、流同步）都值得精读。建议架构师和 MoE 开发者重点关注，了解如何通过小规模 LP 优化专家负载分配。

功能与动机

当前的动态专家调度（dynamic）在路由偏斜数据集（如 GSM8K）上会导致负载不均衡，限制吞吐量。LPLB 通过将每层的令牌分配问题建模为线性规划，并利用所有 EP rank 的全局计数求解最优分配，从而缓解这一瓶颈。

实现拆解

1. JIT CUDA IPM 求解器：在 python/sglang/jit_kernel/lplb/cuda_solver.py 和 csrc/lplb/ipm.cuh 中实现单 SM 融合 IPM 内核，使用 cuBLASDx 的 GEMM 和手写块 Cholesky。通过 load_jit 按形状特化编译，CPU 开销仅 5-10μs。
2. 预处理 / 后处理融合：编写 lp_prep.cuh 和 lp_post.cuh，分别将 8 个和 5 个 torch 操作合并为单内核启动，减少内存流量。cuda_solver.py 中的 prep_lp_inputs 和 extract_log2phy_prob 负责驱动。
3. LPLBSolver 封装：在 python/sglang/srt/eplb/lplb_solver.py 中定义 LPLBSolver 类，初始化时预计算 LP 约束矩阵，每 batch 调用 solve 方法：先对 topk_ids 进行 bincount，再执行 all_reduce 获得全局计数，然后调用求解器输出概率向量 log2phy_prob。设计上所有 rank 独立求解，无需广播。
4. 空 rank 参与保护：修改 python/sglang/srt/layers/moe/topk.py 的 empty_topk_output，在启用 LP 时让空 rank 也调用 solver.solve() 参与 all_reduce，避免死锁。同时修改 deepseek_v2.py 的前向路径，使用统一的空 rank 逻辑。
5. 启动参数与校验：新增 --ep-dispatch-algorithm=lp，在 server_args.py 中添加 check_lplb_server_args 校验（Hopper SM≥9.0、Math-DX 可用、模型架构受支持）。提供 --lplb-require-lp 和 --lplb-require-fused 严格模式确保无静默回退。
6. 分布式测试：添加 test/registered/eplb/test_lplb_distributed.py，使用 torch.multiprocessing.spawn 启动 2 个 CUDA 进程，验证 all-reduce 一致性、空 rank 场景、数值等价于 torch IPM 参考，以及重均衡后求解器正确更新。

关键文件：

- python/sglang/jit_kernel/lplb/cuda_solver.py（模块 JIT 内核；类别 source；类型 core-logic；符号 _sm_ver, _ipm_module, warmup, solve_ipm）：核心 IPM 求解器的 JIT CUDA 实现，包含 solve_ipm、prep_lp_inputs 等关键函数，是 LPLB 的性能基础。
- python/sglang/srt/eplb/lplb_solver.py（模块 均衡调度；类别 source；类型 dependency-wiring；符号 assert_lplb_supported_model, get_global_lplb_solver, set_global_lplb_solver, clear_global_lplb_solvers）：LPLBSolver 类定义，封装 LP 矩阵预计算和每 batch 求解，负责全局计数 all-reduce 和空 rank 处理。
- python/sglang/jit_kernel/lplb/torch_solver.py（模块 求解器入口；类别 source；类型 dependency-wiring；符号 _init_fused_backend, _unavailable_reason, warmup, solve_ipm）：求解器入口模块，按需初始化 fused backend，并在不可用时提供清晰错误信息。
- python/sglang/jit_kernel/lplb/shmem_budget.py（模块 内存预算；类别 source；类型 dependency-wiring；符号 ShmemBreakdown, total_bytes, as_kib, shmem_bytes）：共享内存预算计算，确保 IPM 内核在 GPU 上可部署，否则提示调整问题规模。
- test/registered/eplb/test_lplb_distributed.py（模块 分布式测试；类别 test；类型 test-coverage；符号 _make_metadata, test_dispatch_probability_matches_torch_reference, test_solve_ipm_matches_torch_reference, test_lplb_distributed_two_rank）：多进程分布式测试，覆盖 all-reduce 一致性、空 rank 场景和数值等价性。
- python/sglang/jit_kernel/utils.py（模块 工具函数；类别 source；类型 core-logic；符号 get_mathdx_root, get_mathdx_include_paths）：新增 Math-DX 依赖定位函数，支持环境变量和 pip 包两种方式。
- python/sglang/srt/layers/moe/topk.py（模块 MoE 路由；类别 source；类型 core-logic；符号 empty_topk_output）：修改 empty_topk_output 以支持 LP 空 rank 参与 all-reduce，是死锁修复的关键。
- python/sglang/srt/models/deepseek_v2.py（模块 模型适配；类别 source；类型 data-contract）：修改前向路径，使所有 EP rank 在 LP 模式下参与 all-reduce，消除重复代码。

关键符号：solve_ipm, prep_lp_inputs, extract_log2phy_prob, LPLBSolver.solve, LPLBSolver._solve, assert_lplb_supported_model, empty_topk_output, _init_fused_backend, warmup, get_mathdx_root, get_mathdx_include_paths

关键源码片段

python/sglang/jit_kernel/lplb/cuda_solver.py

核心 IPM 求解器的 JIT CUDA 实现，包含 solve_ipm、prep_lp_inputs 等关键函数，是 LPLB 的性能基础。

```
def solve_ipm(
    A: torch.Tensor, b: torch.Tensor, c: torch.Tensor,
    num_iters: int = DEFAULT_NUM_ITERS,
    result: torch.Tensor | None = None,
) -> torch.Tensor:
```

```
    """Run the fused single-SM IPM kernel using cuBLASDx GEMMs and a hand-written
    block Cholesky for the POSV.  All state is in shared memory."""
    # 确保输入在 CUDA 上且为 float32 类型
    assert A.is_cuda and b.is_cuda and c.is_cuda
    assert A.dtype == torch.float32
    nc, nv = A.shape
    assert b.shape == (nc,), f"b shape mismatch: {b.shape} vs ({nc},)"
    assert c.shape == (nv,), f"c shape mismatch: {c.shape} vs ({nv},)"

    # 编译或从缓存获取指定形状的 IPM 模块
    module = _ipm_module(nc, nv, DEFAULT_BLOCK_DIM, num_iters, _sm_ver())
    # 若未提供输出缓冲区则自动分配（节省 ~20 µs 分配时间）
    if result is None:
        result = torch.empty(nv, dtype=torch.float32, device=A.device)
    # 启动单 block 的内核；所有 LP 状态驻留在共享内存中
    module.ipm_solve(A, b, c, result)
    return result
```

评论区精华

以下是 review 中的核心讨论：

- d_max 累加 bug：gemini-code-assist 指出 IPM 核心中 d_max 被覆盖而非累加，导致步长错误。作者在后续 commit 中修复为 fmaxf 累加。
- 数值稳定性：建议对 Cholesky 对角 fmaxf 避免 NaN，及 d_max 为零时的保护。作者采纳并在 ipm.cuh 中添加了钳位逻辑。
- 启动约束放宽：ch-wan 质疑 --enable-dp-attention 和 --moe-a2a-backend=deepep 是否必需。作者实验后去除了这些强制校验（commit 326b956）。
- 空 rank 逻辑集中：xutizhou 建议将空 rank 参与逻辑移入 empty_topk_output 以减少重复。作者完成重构（commit 0c84e63）。
- Math-DX 依赖：ch-wan 询问 download-mathdx.sh，作者随后移除该脚本，改用 nvidia-mathdx pip 包和 MATHDX_HOME 环境变量（commit edf3217）。
- torch 参考实现：ch-wan 要求提供 torch IPM 参考以比较数值差异。作者添加了 torch_solver.py 中的参考求解器和等价性测试（commit e9dd0dd）。
  - IPM 内核 d_max 累加 bug (correctness): 作者修复为正确的 fmaxf 累加。
  - 数值稳定性改进（Cholesky、步长） (correctness): 作者采纳并添加钳位逻辑。
  - 是否必须 dp-attention 和 deepep (design): 作者实验后去除强制校验，LP 不在强依赖它们。
  - 空 rank 逻辑集中到 empty_topk_output (design): 作者完成重构，统一了 deepseek_v2.py 中的空调用。
  - Math-DX 依赖与下载脚本 (other): 作者移除该脚本，改用 nvidia-mathdx pip 包和 MATHDX_HOME 环境变量。

风险与影响

- 风险：
  1. 新依赖 Math-DX：若 GPU 缺少 cuBLASDx 头文件，服务启动时直接报错，无静默回退；用户需手动安装 nvidia-mathdx 或设置 MATHDX_HOME。
1. GPU 硬件限制：仅支持 Hopper 及以上（SM≥9.0），旧 GPU（如 A100）无法使用；assert_fits 在共享内存超限时也会中断。
2. 模型兼容性有限：仅验证了 DeepSeek-v2 系列等少数架构，其他 MoE 模型如果启用 LP 可能因空 rank 路径不同而触发下游 all_reduce 死锁；assert_lplb_supported_model 在初始化时阻止不支持的模型。
3. JIT 编译时间：首次编译每个形状的 LP 内核需 20-40 秒，已通过预热机制缓解；但若预热失败（如编译错误）会导致启动失败。
4. 空 rank 死锁风险：若新模型引入时未正确处理空 rank 路径，可能导致 all_reduce 挂起；分布式测试覆盖了基本场景，但仍有遗漏可能。
5. 性能回归：在非路由偏斜或预填充受限工作负载上，LP 求解可能带来额外开销，实测 MMLU 和 ShareGPT 的吞吐量变化在±0.5% 内，但不排除更极端场景。 - 影响：
   - 用户：使用 --ep-dispatch-algorithm=lp 需要安装新依赖和 Hopper GPU；不支持其他 MoE 架构。受益用户主要是 DeepSeek 等大模型部署者，在路由偏斜场景下可获 5%+ 吞吐提升。
- 系统：初始化时间增加（JIT 编译 + 矩阵预计算），但默认不启用；运行时每 MoE 层增加约 76μs GPU 求解时间，但通过融合和预分配控制了开销。
- 团队：需维护 4 个新 CUDA 内核文件和 Python 封装；依赖管理新增 Math-DX；需持续扩展模型兼容性列表；分布式测试增加了 CI 资源需求。
- 风险标记：仅支持 Hopper GPU，必须安装 Math-DX，模型兼容性有限，JIT 编译耗时，空 rank 死锁风险

关联脉络

- PR #28404 [AMD][Fix] Skip EPLB topk remap when global server args are unset: 修改了相同的 topk.py 文件，且同为 MoE 专家路由相关调整。