PR #29525 完整报告

sgl-project/sglang

[Feature] Add DeepEPv2 (ElasticBuffer) MoE A2A backend

合并时间：2026-08-20 02:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29525

执行摘要

- 一句话：接入 DeepEPv2 ElasticBuffer，decode 路径 CUDA-graph 可捕获
- 推荐动作：值得精读。核心价值在于三点：一是『固定通信形状换取 CUDA-graph 可捕获性』的架构权衡，这是多节点 MoE decode 优化的可行路径；二是 review 中暴露的测试盲区教训（连续 scale 掩盖 packed UE8M0 stride bug、hidden=512 整除掩盖列布局），对 kernel 测试设计有普适意义；三是声明式 server-args 与 raw runner 校验的时序问题及 resolved_view 解法，是 SGLang 参数体系演进的重要范例。建议关注后续是否有基于 ElasticBuffer 的 TBO/SBO 支持与多节点性能数据补充。

功能与动机

PR body 明确点出核心动机：DeepEP v2 的 ElasticBuffer 是 NCCL 对称内存 all-to-all 引擎，固定 per-rank 容量意味着静态通信形状，使 MoE decode 路径在任何拓扑下（包括 legacy low_latency 不可图化的多节点场景）都可被 CUDA-graph 捕获。因此需要把它作为独立后端 deepep_v2 接入，只替换 expert-parallel 的 dispatch/combine。同时作者强调 accuracy 数据只用于证明与同节点 deepep 基线无系统性偏移（MMLU 500 例标准误约 1.4pp），并非排名。

实现拆解

变更入口是新增 --moe-a2a-backend deepep_v2 与 --deepep-v2-mode direct|hybrid 参数，随后按以下 5 步落地：

1. 参数与校验入口（server_args.py、arg_groups/overrides.py）：新增 deepep_v2 专属参数与 SGLANG_DEEPEP_V2_NUM_MAX_DISPATCH_TOKENS_PER_RANK 容量环境变量；deepep_v2加入声明式a2apasses（_A2A_EP_SPANNING_BACKENDS声明ep_size、_a2a_fusion_adjustments 声明 disable_shared_experts_fusion），并通过 resolved_view(...) 校验最终 moe_runner_backend，避免模型声明（如 mxfp8 + auto -> flashinfer_trtllm）在材料化阶段静默恢复不支持的 runner；boot 时校验 chunked-prefill 预算不超过 per-rank 容量，TBO/SBO 与强制共享 expert 融合在启动时拒绝。
2. dispatcher 核心（新文件 layers/moe/token_dispatcher/deepep_v2.py，+548 行）：DeepEPv2Buffer.get_buffer 按进程组 key 缓存单例 ElasticBuffer；布局按 推理阶段而非通信模式选择——decode 用展开布局（expanded）+ masked GEMM（静态形状、graph-capturable，hybrid 模式下也可用），extend/prefill 用连续布局（contiguous）；只暴露 single-shot dispatch()/combine() 与严格 handle 生命周期；导入守卫先于

DeepEP import 检查，未安装 deep_ep 时模块仍可干净导入。

3. Triton repack 内核（kernels/ops/moe/ep_moe_kernels.py，+402 行）：新增 expand_to_masked_slab / masked_slab_to_expand / ep_scatter_from_psum / ep_expand_init_m_indices_from_psum 等内核；固定 per-expert worker pool 保证 grid 静态以图安全；top-k 权重核内融合；scale 拷贝按源 pack-dim stride 处理，兼容 Blackwell 的列主序 packed-UE8M0 与 Hopper 的行主序 fp32。
4. runner 适配（layers/moe/moe_runner/deep_gemm.py，+218/-1）：注册 pre_permute_deepep_v2_to_deep_gemm / post_permute_deep_gemm_to_deepep_v2，消费 DeepEP 的 128 对齐 psum 作为 masked/contiguous GEMM 的 group offset；deepep_v2 只支持 deep_gemm runner，MoeRunner 对不匹配 runner 也拒绝；layers/moe/utils.py 新增 MoeA2ABackend.DEEPEP_V2、is_deepep_v2、DeepEPv2Fp8ScaleFormat 与 get_deepep_v2_fp8_scale_format（scale 布局按 DeepGEMM JIT 配置随硬件变化）。
5. 外围修复与测试配套：state_capturer/routed_experts.py 将 deepep_v2 归类为 scattered-a2a 后端（_is_scattered_a2a_backend()），避免 DP>1 时读到未写的 buffer 行返回垃圾 expert id；token_dispatcher/base.py 新增 DispatchOutputFormat.DEEPEP_V2 与判定辅助；测试配套包括 masked-slab roundtrip 单元测试（含空 expert、溢出 fail-fast、生产 packed-UE8M0 scale、CUDA-graph 捕获回放）、server-args 解析测试（runner 解析 /reject/ 声明恢复 fail-fast）、capturer 分类测试，以及 DP>1 的 routed-experts readback e2e 测试。

关键文件：

- python/sglang/srt/layers/moe/token_dispatcher/deepep_v2.py（模块 专家路由；类别 source；类型 core-logic；符号 DeepEPv2DispatchOutput, DeepEPv2CombineInput, DeepEPv2Buffer, get_buffer）：新增的 deepep_v2 dispatcher 主文件（548 行），实现 ElasticBuffer 单例管理、按推理阶段选布局、single-shot dispatch/combine 与导入守卫，是整个后端的中枢。
- python/sglang/srt/layers/moe/moe_runner/deep_gemm.py（模块 MoE 执行；类别 source；类型 core-logic；符号 pre_permute_deepep_v2_to_deep_gemm, post_permute_deep_gemm_to_deepep_v2, DeepGemmRunnerInput）：deepep_v2 唯一支持的 runner 适配层：新增 pre/post permute 将 FP8 激活 +scale 转成 DeepGemmRunnerInput，并复用 DeepEP 128 对齐 psum 作为 group offset；同时为共用 DeepGEMM 核心的零 token 短路做了一次全局回退。
- python/sglang/kernels/ops/moe/ep_moe_kernels.py（模块 Triton 内核；类别 infra；类型 infrastructure；符号 _fwd_kernel_ep_scatter_psum_init, ep_scatter_from_psum, _fwd_kernel_ep_expand_m_indices_init, ep_expand_init_m_indices_from_psum）：新增 402 行 Triton repack 内核，是 decode 热路径的几何转换核心；固定 per-expert worker pool 保证 grid 静态以图安全，scale 拷贝按 pack-dim stride 处理兼容 Blackwell packed-UE8M0。
- python/sglang/srt/layers/moe/utils.py（模块 工具层；类别 source；类型 core-logic；符号 is_deepep_v2, DeepEPv2Fp8ScaleFormat, get_deepep_v2_fp8_scale_format）：MoE 后端与 runner 的公共契约层：新增 DEEPEP_V2 枚举、is_deepep_v2、

DeepEPv2Fp8ScaleFormat 与按硬件解析 scale 布局的函数，供 dispatcher 与 runner 共用。

- python/sglang/srt/server_args.py（模块 参数解析；类别 source；类型 core-logic）：参数解析入口：新增 deepep_v2 分支、auto runner 解析、resolved_view 声明态校验、chunked-prefill 容量检查与 TBO/SBO 拒绝，是后端可用性的第一道防线。
- test/registered/unit/layers/moe/test_deepep_v2_masked_slab.py（模块 内核测试；类别 test；类型 test-coverage；符号 TestDeepEPv2MaskedSlab, _build_layout, _check_expand_roundtrip, expand_to_masked_slab）：针对 masked-slab repack 内核的单元测试，覆盖 review 点名的空 expert、溢出 fail-fast、top-k 权重融合与真实 packed-UE8M0 scale；是 Blackwell 回归的 CI 防线。
- python/sglang/srt/state_capturer/routed_experts.py（模块 状态捕获；类别 source；类型 core-logic；符号 _is_scattered_a2a_backend）：修复 review 指出的关键正确性缺口：deepep_v2 未被 is_deepep() 识别时，DP attention + --enable-return-routed-experts 会让 dp_rank>0 读到未写的 buffer 行；改为 _is_scattered_a2a_backend() 统一分类。
- test/registered/ep/test_routed_experts_dp_readback.py（模块 E2E 测试；类别 test；类型 test-coverage；符号 TestRoutedExpertsReadbackDeepEPv2, _ReadbackMixin, _one_request, test_dp2_readback）：DP>1 readback e2e 测试：用 solo-vs-concurrent oracle 验证 deepep_v2 在 DP attention 下读回正确 expert id，是捕获分类回归的端到端防线。
- test/registered/unit/state_capturer/test_routed_experts_scattered_a2a.py（模块 分类测试；类别 test；类型 test-coverage；符号 TestScatteredA2ABackendHelper, test_classification, TestGetLocalSliceBackendBranch, test_deepep_v2_reads_buffer_head）：capturer 分类与 buffer 切片分支的单元测试，锁定 deepep_v2 与 deepep 读取 buffer 头的行为一致，防止回归到按全局 DP offset 读脏数据。
- python/sglang/srt/layers/moe/token_dispatcher/base.py（模块 契约定义；类别 source；类型 core-logic；符号 format_is_deepep_v2, is_deepep_v2）：数据契约扩展：DispatchOutputFormat / CombineInputFormat 新增 DEEPEP_V2 枚举与 format_is_deepep_v2 / is_deepep_v2 判定，是 dispatcher 输出被 runner 分发的依据。

关键符号：DeepEPv2Buffer.get_buffer, _ensure_deepep_v2_available, _ensure_fp8_quant_available, _quantize_for_deepep_v2_dispatch, pre_permute_deepep_v2_to_deep_gemm, post_permute_deep_gemm_to_deepep_v2, expand_to_masked_slab, masked_slab_to_expand, ep_scatter_from_psum, ep_expand_init_m_indices_from_psum, is_deepep_v2, get_deepep_v2_fp8_scale_format, _is_scattered_a2a_backend

关键源码片段

python/sglang/srt/layers/moe/moe_runner/deep_gemm.py

deepep_v2 唯一支持的 runner 适配层：新增 pre/post permute 将 FP8 激活 +scale 转成 DeepGemmRunnerInput，并复用 DeepEP 128 对齐 psum 作为 group offset；同时为共用 DeepGEMM 核心的零 token 短路做了一次全局回退。

```
# python/sglang/srt/layers/moe/moe_runner/deep_gemm.py
# deepep_v2 -> deep_gemm 的 pre-permute 适配：把 ElasticBuffer 的
# dispatch 输出转成 DeepGemmRunnerInput，按展开 / 连续两种布局分流。

@register_pre_permute("deepep_v2", "deep_gemm")
def pre_permute_deepep_v2_to_deep_gemm(
    dispatch_output: DeepEPv2DispatchOutput,
    quant_info: DeepGemmMoeQuantInfo,
    runner_config: MoeRunnerConfig,
    running_state: dict,
) -> DeepGemmRunnerInput:
    from sglang.kernels.ops.moe.ep_moe_kernels import (
        ep_expand_init_m_indices_from_psum,
        ep_scatter_from_psum,
    )

    hidden_states = dispatch_output.hidden_states
    hidden_states_scale = dispatch_output.hidden_states_scale
    topk_ids = dispatch_output.topk_ids
    topk_weights = dispatch_output.topk_weights
    psum_num_recv_tokens_per_expert = dispatch_output.psum_num_recv_tokens_per_expert
    is_expanded = dispatch_output.is_expanded
    hidden_states_scale_tma_aligned = dispatch_output.hidden_states_scale_tma_aligned
    deepep_v2_use_masked = dispatch_output.use_masked_gemm
    deepep_v2_expected_m = dispatch_output.expected_m
    deepep_v2_masked_max_m = dispatch_output.masked_max_m
    deepep_v2_total_expanded = dispatch_output.total_expanded
    deepep_v2_expert_alignment = dispatch_output.expert_alignment

    # deepep_v2 只支持 deep_gemm runner：dispatch 输出 FP8 激活 + scale，
    # 缺少 scale 说明走了错误的 runner/ 量化组合，直接 fail fast。
    if hidden_states_scale is None:
        raise RuntimeError(
            "DeepEP v2 -> DeepGEMM requires FP8 dispatch output with activation "
            "scales, but the dispatch output carried none."
        )
    assert runner_config.activation == "silu"

    if is_expanded:
        # decode 路径：展开布局下按 psum 重建 m_indices（静态形状、graph-safe），
        # 随后 repack 成 [E_local, max_m, hidden] masked slab，使 GEMM 计算量
        # 只与每个 expert 的真实行数（masked_m）成正比，而非 padded max_m。
        if psum_num_recv_tokens_per_expert is None:
            raise RuntimeError(
                "DeepEP v2 requires the native expert prefix sums from the "
                "ElasticBuffer dispatch handle."
            )
        all_tokens = hidden_states.shape[0]
        running_state["all_tokens"] = all_tokens
```

```
        running_state["hidden_states_shape"] = hidden_states.shape
        running_state["hidden_states_device"] = hidden_states.device
        running_state["hidden_states_dtype"] = hidden_states.dtype
        running_state["topk_ids"] = None
        running_state["topk_weights"] = topk_weights
        running_state["deepep_v2_expanded"] = True
        # 后续调用 ep_expand_init_m_indices_from_psum 与 expand_to_masked_slab,
        # 并把 masked GEMM 元数据（expected_m / masked_max_m / expert_alignment）
        # 填入 DeepGemmRunnerInput；topk 权重在 repack 阶段核内融合。
    else:
        # extend/prefill 路径：连续布局，ep_scatter_from_psum 按 psum 计算
        # 偏移写回，128 对齐契约保证 psum 可作 contiguous GEMM 的 group offset。
        ...
```

test/registered/unit/layers/moe/test_deepep_v2_masked_slab.py

针对 masked-slab repack 内核的单元测试，覆盖 review 点名的空 expert、溢出 fail-fast、top-k 权重融合与真实 packed-UE8M0 scale；是 Blackwell 回归的 CI 防线。

```
# test/registered/unit/layers/moe/test_deepep_v2_masked_slab.py
# 覆盖 review 中点名的边界：空 expert、单个热点 expert、per-expert 计数
# 接近 / 超过 max_m（溢出必须 fail fast 而非静默截断）、top-k 权重只在真实
# 行上融合、expanded <-> masked_x 布局往返、FP8 激活 + scale 路径。

def _build_layout(counts, align, hidden, dtype, with_scale=False, scale_hidden=4):
    """按 per-expert 计数构造 DeepEP v2 展开布局：
    expert e 占据 [align(psum[e-1]), psum[e]) 行，psum[-1] == 0。"""
    starts, psum = [], []
    prev_end = 0
    for c in counts:
        start = ((prev_end + align - 1) // align) * align
        end = start + c
        starts.append(start)
        psum.append(end)
        prev_end = end
    total = max(((prev_end + align - 1) // align) * align, 1)

    # 每个真实行取唯一值（保持在 e4m3 范围内），hidden 维上交替 x1/x2：
    # 能抓住「整行广播某列」或「hidden 偏移错位」的内核实现错误。
    base = torch.zeros((total, hidden), dtype=torch.float32, device=DEVICE)
    col_gain = 1.0 + (torch.arange(hidden, device=DEVICE) % 2).float()
    for s, c in zip(starts, counts):
        for j in range(c):
            base[s + j] = float((s + j) % 200 + 1) * col_gain
    recv_x = base.to(dtype)

    scale = None
    if with_scale:
        # review 教训：若用连续 FP32 scale 构造，无法回归 Blackwell 的
        # packed UE8M0 列主序 stride；且 hidden=512 恰好整除 packed 列数
```

```
            # 会进一步掩盖错误，因此生产版测试改用真实 quantizer 构造。
            scale = torch.zeros((total, scale_hidden), dtype=torch.float32, device=DEVICE)
            col = torch.arange(scale_hidden, dtype=torch.float32, device=DEVICE)
            for s, c in zip(starts, counts):
                for j in range(c):
                    scale[s + j] = float((s + j) % 50 + 1) * 0.5 + col

        psum_t = torch.tensor(psum, dtype=torch.int32, device=DEVICE)
        return recv_x, scale, psum_t, starts, total


    def _check_expand_roundtrip(self, counts, dtype, with_scale, topk=False):
        # 展开 -> masked slab -> 展开的往返校验：masked_m 必须等于真实计数，
        # 真实行逐元素相等；top-k 模式同时验证权重只在真实行上融合。
        recv_x, scale, psum, starts, total = _build_layout(
            counts, self.ALIGN, self.HIDDEN, dtype, with_scale=with_scale
        )
        E = len(counts)
        masked_x, masked_x_scale, masked_m = expand_to_masked_slab(
            recv_x, scale, psum, E, self.MAX_M, self.ALIGN
        )
        self.assertEqual(masked_m.tolist(), list(counts))
        self.assertEqual(tuple(masked_x.shape), (E, self.MAX_M, self.HIDDEN))
        for e, (s, c) in enumerate(zip(starts, counts)):
            for j in range(c):
                torch.testing.assert_close(masked_x[e, j].float(), recv_x[s + j].float())
        # ... round-trip 回展开布局并校验 top-k 权重融合结果
```

评论区精华

核心讨论围绕四类问题：

1. 内核正确性：gemini-code-assist 指出 _fwd_kernel_ep_scatter_psum_init 无边界掩码会导致并发 program 越界写坏后续 expert 的 m_indices（critical），作者在 commit 36a0653 用 idx < cur_end 掩码修复。
2. 设计边界：ch-wan 质疑 parallel_state.py 的 device_id 改动与 deepep_v2 无关，作者撤回该改动，改为在 deepep_v2 路径内设置 EP_REUSE_NCCL_COMM=0 绕开 DeepEP 对 device-bound PG 的要求。liz-badada 指出 v2 校验 raw runner 与声明式 server-args 材料化存在时序冲突，作者改用 resolved_view(...) 校验最终声明状态并新增 fail-fast 测试。
3. 测试盲区：liz-badada 指出 Blackwell stride 修复后测试仍用连续 FP32 scale，无法回归 packed UE8M0 布局；作者改用生产 quantizer 构造 packed scale，并意外发现 hidden=512 恰好整除 packed 列数会掩盖该问题，测试因此改用非整除 hidden。
4. 可用性与验证：laixinn 反馈严格按文档启动时 EP16 报 symmetric memory 错误，作者澄清需 NCCL_CUMEM_ENABLE=1 或 --enable-symm-mem 并列明副作用；yh0903 建议补充多节点 EP A2A e2e/perf，作者补充了多节点配置说明与 three-question gate 通过记录，但未给出多节点性能对比表。

- _fwd_kernel_ep_scatter_psum_init 无边界掩码的越界写风险 (correctness): 已修复：commit 36a0653 给尾块加 mask（idx < cur_end），并补了对应回归验证。
- gather_out 用 torch.empty 是否引入 NaN 传播 (correctness): 维持 torch.empty，注释更新为「只写真实行，padding 未初始化且永不读取」。
- parallel_state.py 的 device_id 改动是否与 PR 相关 (design): 撤回 parallel_state.py 改动，共享初始化不受影响。
- is_deepep() 不识别 deepep_v2 导致 DP>1 读回脏数据 (correctness): 修复：三个 gate 改走 _is_scattered_a2a_backend()（deepep ∨ deepep_v2），审计其余调用点无 v2 相关遗漏；新增 solo-vs-concurrent oracle 的 DP>1 e2e 测试。
- Blackwell packed UE8M0 scale 缺少真实回归测试 (testing): 测试升级为生产 packed scale + 真实捕获回放；B200/B300 端到端复验通过（B300 GSM8K 0.958 / MMLU 0.899）。
- deepep_v2 修改裸 runner 与声明式 server-args 材料化的时序冲突 (design): 已修复，新增 test_runner_restored_by_declaration_fails_fast 测试。
- DeepEP v2 是否应自动启用 symmetric memory (question): 文档层面解决：明确两种启用方式及副作用；未做自动启用。
- 多节点 EP A2A e2e / 性能验证缺失 (testing): 部分解决：多节点功能通过，但未提供多节点性能对比表。

风险与影响

- 风险：
  1. decode 热路径正确性：masked-slab repack 内核运行在每步 decode 上，边界掩码与 per-expert 容量上限依赖 masked_max_m 计算正确；测试覆盖了溢出 fail-fast，但生产环境动态路由变化大，仍有回归风险。
  2. NCCL 对称内存依赖：DeepEP v2 必须在 NCCL_CUMEM_ENABLE=1（或 --enable-symm-mem）下运行，否则启动即报 Communicator does not support symmetric memory!；--enable-symm-mem 还会额外启用 NVLS 与 4 GB 预分配，影响显存规划。
  3. server-args 分支扩大：server_args.py 新增 101 行校验分支，曾因缺 NS("exec.moe") marker 触发 test_server_args_namespaces 失败；参数解析是全局入口，回归影响所有启动路径。
  4. Blackwell 特有路径无 CI 覆盖：UE8M0 与 TMA 对齐只在 B200/B300 上有效，CI 无 Blackwell 硬件，长期依赖人工回归。
  5. 容量配置：SGLANG_DEEPEP_V2_NUM_MAX_DISPATCH_TOKENS_PER_RANK 必须覆盖 per-rank chunk 且为 page_size 倍数，配置错误依赖 boot 时校验，运行时调整成本高。 - 影响：对用户侧：DeepSeek-V4-Flash 等使用 ep + DP attention 的 MoE 模型在多节点场景首次获得 CUDA-graph 可捕获的 EP A2A decode 路径，H20 上 prefill 吞吐提升约 3.2%、decode 持平（-0.9%）；但使用门槛升高（需理解 symmetric memory 与容量配置）。对系统侧：新增一个 A2A 后端与约 2000 行代码，MoeA2ABackend、DispatchOutputFormat 等枚举契约扩展，is_deepep() 语义审计为

后续类似后端（mooncake/mori）提供参照。对团队侧：与 PR 29402 存在路线重叠，需要协调；新后端默认 inert，未安装 DeepEP 不影响现有用户。 - 风险标记：decode 热路径新增 Triton 内核，NCCL 对称内存配置依赖，Blackwell 特有路径无 CI 覆盖，容量配置错误仅启动时校验，新增参数分支影响全局启动解析

关联脉络

- PR #29402 同主题平行 PR（讨论中提及，标题不在本次材料内）：yh0903 评论中明确提到其团队也在推进同一 DeepEP v2 / ElasticBuffer 集成；本 PR 曾用 epv2 命名，后重命名为 deepep_v2 与之对齐。
- PR #35294 [NIXL] Query EP top-k index dtype: 同为 MoE token dispatcher / a2a 后端维护线，反映 SGLang MoE 通信后端家族（deepep / deepep_v2 / nixl / mooncake）持续演进与参数收敛。