PR #31987 完整报告

sgl-project/sglang

[BCG][3/N] Enable bcg on dsa & deepep a2a backend

合并时间：2026-08-01 07:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31987

执行摘要

- 一句话：BCG 预填图扩展至 DSA 模型与 DeepEP a2a MoE
- 推荐动作：值得精读，属于 BCG 体系的关键里程碑。三个设计决策有借鉴价值：capture stub + barrier 让 rank 耦合集体通信安全进入 CUDA graph 捕获；单一回放资格谓词同时服务 forward 与 schedule 两个时点，消除 DP collective 错配隐患；with_output 变体（调用方在图内分配输出、break 只 mutate）避免 per-shape 桥接张量。merrymercy 的 review 完整驱动了一次“条件收口 + 形态收敛”的重构，可作为大型 PR 协作范本。阅读时建议对照 commit 序列中的 A/B 实验记录（seeded dummies 引入又撤销、decode-first 顺序引入又撤销、内存预留 4 轮校准），这些是难得的工程决策素材。

功能与动机

PR body 为空模板，动机来自 commit 与 review。核心三点：其一，_disable_breakable_cudagraph_if_incompatible 中 blanket 的“MLA attention”规则早于 DSA BCG 埋点，把本可支持的 DSA-backed MLA 一并禁用——DSA 在 BCG 下强制 use_mha=False 走稀疏路径，其 indexer 已具备 eager 分割能力（bcg_dsa_indexer_prefill_split）；其二，MoE 在图内捕获只能用 low-latency a2a 模式，prefill 载荷下带宽极差，实测 GLM-5.2-FP8 tp8/dp8/deepep gsm8k 400q 图内捕获 105.9s 对 段间 eager 48.7s；其三，上游 #31682 默认开启 DP 下 breakable prefill 后，mlp-sync 共识只检查 forward mode，rank 本地条件（prefix、token 上限、padding 因子、embeds）不一致时单个 rank 静默退化 eager，DSA 上 eager rank 会重新推导 use_mha=True 走 MHA 路径，collective 错配挂死。

实现拆解

- _disable_breakable_cudagraph_if_incompatible 的“MLA attention”规则从 blanket 改为 use_mla_backend() and not is_deepseek_dsa(...)；“MoE A2A backend”规则改为仅放行 none 与 deepep（pplx 等其他 a2a 后端保持 BCG 禁用，等后续验证）；新增“two-batch overlap”禁用规则——TBO 的 capture_prepare 会在 dummy batch 上 assert，冻结的 split index 在 replay 也必然错误。
- _handle_cuda_graph_config 插入新方法 _apply_deepep_adjustments：DeepEP + breakable 时把 prefill bucket 对齐到 8 的倍数（非 8 倍数会确定性挂死 DeepEP a2a 捕获，根因未定位，先规避），bs 为 None 时用 max_bs or 2048 物化默认 ladder；对齐结果同步写回 max_bs。

- reserve_for_graph_mb 增加 1 GB 的 prefill-BCG DeepEP 增量预留（桥接池 + NVL first-touch），并把经过多轮实测校准的经验值（8GB -> 6.5GB -> 1.5GB + 1GB）以命名清晰的项沉淀；顺带把 resolved_view 提升为模块级 import，删除约 10 处函数内 import（review 要求）。
- 新增 _a2a_forward_with_output_impl：临时置 is_extend_in_batch=True（prefill 图 runner 为 captured-LL 路径把它钉死在 False，而段间 eager 需要 NORMAL 模式）后执行完整 forward_impl，结果 copy_ 进调用方预分配的输出张量；新增 _a2a_forward_capture_stub：捕获期仅 output.zero_()，跳过 rank 耦合的 a2a。
- 二者经 eager_on_graph(True, capture_stub=...) 包装为类方法 a2a_forward_with_output；forward 在 is_in_breakable_cuda_graph() 分支中于捕获区域内分配 torch.empty_like 输出并调用它，返回 None 契约让 eager_on_graph 无需为每个 shape/ 层强引用桥接张量。
- 分割节点从 deepseek_v2.forward_deepep 上移至此（模型无关），任何 a2a MoE 模型在默认 BCG 下都不会再走 low-latency 容量断言路径。
- eager_on_graph 新增 capture_stub 参数：捕获期用 stub 替代真实 body，只记录输出地址；BreakableCUDAGraphCapture 新增 barrier_fn，在 _end_current_segment 之后、break body 之前执行，吸收段拆卸带来的 rank 抖动（DeepEP NORMAL dispatch 有 100s CPU 硬超时，是编译期常量，无法加长）。replay 绕过 wrapper，不产生额外同步。
- PrefillCudaGraphRunner 新增 keyword-only 的 can_replay_locally：承载全部 rank 本地条件（Full 槽位、LoRA 资格、embeds、MHA-companion prefix 与 DSA 豁免、target verify、hidden mode、logprob、token 上限与 padding 因子、chunked prefix 不可捕获），并通过 dsa_sparse_prefill_forced 类级默认值兼容 __new__ 构造的测试实例。
- can_run_graph 变成薄适配：DP 共识门（global_num_tokens_cpu + can_run_dp_breakable_cuda_graph）+ 非活跃 rank 门 + 本地谓词；原 _has_unsupported_mha_prefix 与重复逻辑删除。
- dp_attn.prepare_mlp_sync_batch_raw 新增 model_runner 参数，调度期投票直接用 ScheduleBatch 字段调用同一谓词；MLPSyncBatchInfo 把单一 can_cuda_graph 拆成 can_run_decode_cuda_graph 与 can_run_prefill_cuda_graph 两个投票位（打包列布局不变，语义按阶段命名，尾部追加）。
- scheduler.init_dp_attn_adapter 通过 BaseSpecWorker.target_worker 显式解析目标模型 runner，修复 EAGLE/MTP 场景 spec worker 无 model_runner 属性导致投票静默 permissive 的缺口。
- deepseek_v2.forward_deepep：BCG 下 shared experts 的 alt-stream 双流重叠失效（事件录制与等待会跨捕获段，DSV4-Flash FP4 在 16-token bucket 出现 token salad），改为段内 torch.cuda.current_stream().wait_event(shared_event) 即刻汇合，保留 record_stream 标记防止分配器在 break 处回收 shared_output。
- eagle_worker_v2.py / eagle_worker_common.py / multi_layer_eagle_worker_v2.py：can_cuda_graph 改名 can_run_decode_cuda_graph，与新增 prefill 投票位语义对齐（纯改名，无行为变化）。
- 测试配套偏薄：test_prefill_cuda_graph_padding.py 仅为 ForwardBatch mock 补 extend_prefix_lens_cpu；benchmark/one_batch.py 增加对应配置键。

关键文件：

- python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py（模块 预填图回放；类别 source；类型 data-contract；符号 _has_unsupported_mha_prefix, can_run_graph, can_replay_locally）：回放资格判定单一化与 DSA 豁免的核心文件：新增 can_replay_locally 作为 forward 时点与 DP 调度投票共用的唯一事实来源，can_run_graph 退化为薄适配，并修复 __new__构造测试实例的属性缺失问题。
- python/sglang/srt/layers/moe/ep_moe/layer.py（模块 MoE 层；类别 source；类型 core-logic；符号 _a2a_forward_with_output_impl, _a2a_forward_capture_stub）：DeepEP a2a 分割节点的落点：把 dispatch -> experts -> combine 整段作为 BCG 段间 eager break，配合 capture stub 跳过捕获期 rank 耦合通信，是 prefill 下 NORMAL 模式性能收益（105.9s -> 48.7s）的直接来源。
- python/sglang/srt/server_args.py（模块 配置解析；类别 source；类型 core-logic；符号 _apply_deepep_adjustments）：兼容规则收窄与 DeepEP 专属配置调整：MLA 规则排除 DSA、a2a 规则仅放行 DeepEP、新增 TBO 禁用，_apply_deepep_adjustments 将 prefill bucket 对齐到 8 的倍数，并校准 1 GB 增量显存预留。
- python/sglang/srt/managers/scheduler_components/dp_attn.py（模块 DP 注意力；类别 source；类型 dependency-wiring）：DP 共识投票改造：MLPSyncBatchInfo 拆分为 decode/prefill 两个投票位，调度期直接调用 prefill runner 的 can_replay_locally，使各 DP rank 在调度时点就达成一致的回放决策。
- python/sglang/srt/model_executor/runner_backend_utils/breakable_cuda_graph/breakable_cuda_graph.py（模块 图捕获器；类别 source；类型 data-contract；符号 eager_on_graph）：捕获机制增强的核心基础设施：eager_on_graph 新增 capture_stub 参数，BreakableCUDAGraphCapture 新增 barrier_fn，让 rank 耦合集体通信（DeepEP 100s CPU 超时）能在捕获期安全执行。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 dependency-wiring）：DP 适配器接线修复：通过 BaseSpecWorker.target_worker 解析目标模型 runner 注入 SchedulerDPAttnAdapter，修复 EAGLE/MTP 场景投票静默 permissive 的缺口。
- python/sglang/srt/models/deepseek_v2.py（模块 模型层；类别 source；类型 data-contract）：BCG 下 shared experts 双流重叠的正确性修复：跨捕获段录制与等待 event 会污染输出（DSV4-Flash token salad），改为段内立即汇合并保留 record_stream 标记。
- test/registered/unit/model_executor/runner/test_prefill_cuda_graph_padding.py（模块 预填测试；类别 test；类型 test-coverage）：测试配套：为 ForwardBatch mock 补 extend_prefix_lens_cpu 字段，适配 can_run_graph 无条件向 can_replay_locally 传该字段的新契约。

关键符号：can_replay_locally, can_run_graph, _a2a_forward_with_output_impl, _a2a_forward_capture_stub, a2a_forward_with_output, eager_on_graph, _apply_deepep_adjustments, prepare_mlp_sync_batch_raw, _disable_breakable_cudagraph_if_incompatible, forward_deepep

关键源码片段

python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py

回放资格判定单一化与 DSA 豁免的核心文件：新增 can_replay_locally 作为 forward 时点与 DP 调度投票共用的唯一事实来源，can_run_graph 退化为薄适配，并修复 __new__构造测试实例的属性缺失问题。

```
# python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py
# can_replay_locally 是 rank 本地回放资格的唯一事实来源：
# can_run_graph（forward 时点）与 dp_attn 的 mlp-sync 投票（调度时点）
# 都是它的薄适配。所有 DP rank 必须对 replay-vs-eager 达成一致，
# 否则 collective 会错配。

class PrefillCudaGraphRunner(BaseCudaGraphRunner):
    # DSA 在 BCG capture/replay 中强制 use_mha=False 走稀疏路径，
    # 稀疏路径通过 device 侧元数据接受任意 prefix，因此豁免 MHA-prefix
    # 禁令；类级默认 False 让 __new__ 构造的测试实例保持保守行为。
    dsa_sparse_prefill_forced: bool = False

    def can_replay_locally(
        self, *, batch_size, num_tokens, input_embeds, replace_embeds,
        prefix_lens, is_target_verify, capture_hidden_mode, return_logprob,
        lora_ineligible=False, chunked_prefix_uncapturable=False,
    ) -> bool:
        if self._is_full_backend and batch_size > self._capture_req_slots:
            return False
        if lora_ineligible or input_embeds is not None or replace_embeds is not None:
            return False
        # 带 prefix 的 batch 会走 MHA companion 路径，其捕获状态是
        # 冻结的 prefix-free；DSA 模型豁免（见上面的类注释）。
        if (self.prefill_backend_name == Backend.BREAKABLE
                and self.has_mha_companion_layers
                and not self.dsa_sparse_prefill_forced
                and prefix_lens is not None and any(prefix_lens)):
            return False
        # FullCG 的 chunked-prefix 拓扑只覆盖有界 prefix，该标志只对
        # FULL 后端生效，对 breakable 投票路径是惰性的。
        if chunked_prefix_uncapturable or is_target_verify:
            return False
        if (capture_hidden_mode is not None
                and capture_hidden_mode != self.capture_hidden_mode):
            return False
        if return_logprob and not self._uses_eager_prefill_tail():
            return False
        if num_tokens is None:
            return True
        if num_tokens > self.max_num_tokens:
            return False
```

```
        # load_batch 会按 bucket 补齐，这里只拒绝补丁过度的浪费。
        padded = self._pad_to_bucket(num_tokens, self.capture_num_tokens)
        if padded > num_tokens * _MAX_PREFILL_CUDA_GRAPH_PADDING_FACTOR:
            return False
        return True
```

python/sglang/srt/layers/moe/ep_moe/layer.py

DeepEP a2a 分割节点的落点：把 dispatch -> experts -> combine 整段作为 BCG 段间 eager break，配合 capture stub 跳过捕获期 rank 耦合通信，是 prefill 下 NORMAL 模式性能收益（105.9s -> 48.7s）的直接来源。

```
# python/sglang/srt/layers/moe/ep_moe/layer.py
# DeepEPMoE 在 breakable CUDA graph 下的 eager 分割节点：
# dispatch -> experts -> combine 整段在捕获段之间以 NORMAL 模式重跑，
# 规避 low-latency 模式在 prefill 载荷下的带宽劣势。

class DeepEPMoE(FusedMoE):
    def _a2a_forward_with_output_impl(
        self, hidden_states, topk_weights, topk_ids, router_logits, output
    ) -> None:
        # prefill 图 runner 会把 is_extend_in_batch 钉死在 False（PCG 捕获
        # low-latency 路径需要），而此处 MoE 在段间 eager 运行，必须临时恢复
        # True，让 DeepEP 解析到 NORMAL 模式的 a2a。
        saved_is_extend_in_batch = get_is_extend_in_batch()
        set_is_extend_in_batch(True)
        try:
            output.copy_(
                self.forward_impl(
                    hidden_states,
                    StandardTopKOutput(topk_weights, topk_ids, router_logits),
                )
            )
        finally:
            set_is_extend_in_batch(saved_is_extend_in_batch)

    def _a2a_forward_capture_stub(
        self, hidden_states, topk_weights, topk_ids, router_logits, output
    ) -> None:
        # 捕获期只记录输出 buffer 地址，跳过 rank 耦合的 a2a 集体通信，
        # 避免 100s CPU 硬超时；warmup 与 replay 才执行真实 body。
        output.zero_()

    # with_output 契约：output 由调用方在捕获区域内分配（graph pool 存储，
    # 跨 shape/ 层复用），break 只做 in-place 修改，eager_on_graph 无需为
    # 每个 shape 保留强引用的桥接张量。
    a2a_forward_with_output = eager_on_graph(
        True, capture_stub=_a2a_forward_capture_stub
    )(_a2a_forward_with_output_impl)
```

```
    def forward(self, hidden_states, topk_output):
        # DeepEP NORMAL 模式不可捕获，BCG 下将它作为 eager 节点执行。
        if is_in_breakable_cuda_graph():
            assert TopKOutputChecker.format_is_standard(topk_output), (
                'Only standard topk output is supported for breakable cuda graph'
            )
            output = torch.empty_like(hidden_states)
            self.a2a_forward_with_output(
                hidden_states,
                topk_output.topk_weights,
                topk_output.topk_ids,
                topk_output.router_logits,
                output,
            )
            return output
        # tc_piecewise 分支（moe_forward_piecewise_cuda_graph_impl）与
        # 普通 forward_impl 分支保持不变。
        if is_in_tc_piecewise_cuda_graph():
            return moe_forward_piecewise_cuda_graph_impl(
                hidden_states, topk_output.topk_weights, topk_output.topk_ids,
                topk_output.router_logits, self.layer_id,
            )
        return self.forward_impl(hidden_states, topk_output)
```

python/sglang/srt/model_executor/runner_backend_utils/breakable_cuda_graph/breakable_cuda_graph.py

捕获机制增强的核心基础设施：eager_on_graph 新增 capture_stub 参数，BreakableCUDAGraphCapture 新增 barrier_fn，让 rank 耦合集体通信（DeepEP 100s CPU 超时）能在捕获期安全执行。

```
# python/sglang/srt/model_executor/runner_backend_utils/breakable_cuda_graph/breakable_
cuda_graph.py
# eager_on_graph：把函数转成捕获段之间的 eager break 节点。
# 段拆卸（end segment + instantiate）是捕获中最慢、rank 间抖动最大的步骤，
# 因此先经过 barrier 再进入可能含 rank 耦合集体通信的 break body。

def eager_on_graph(enable: bool, capture_stub: Optional[Callable] = None):
    def decorator(inner: Callable):
        if not enable:
            return inner

        def wrapper(*args, **kwargs):
            capture = _current_capture_var.get()
            if capture is None:
                return inner(*args, **kwargs)

            # 结束当前捕获段，后续代码进入新段。
            capture._end_current_segment()
```

```
            # barrier 吸收段拆卸造成的 rank 抖动，避免 DeepEP NORMAL dispatch
            # 100s CPU 硬超时被误触发；仅捕获期执行，replay 走 replay_fn。
            if capture._barrier_fn is not None:
                capture._barrier_fn()

            # 捕获期可用 capture_stub 替代真实 body：内容不会被消费，
            # 只记录输出地址；warmup 与 replay 跑真实 inner。
            if capture_stub is not None:
                output = capture_stub(*args, **kwargs)
            else:
                output = inner(*args, **kwargs)

            # 输入张量转 weak ref（storage 由段 CUDAGraph 的 mempool 保活），
            # 输出是捕获间隙分配的静态输入地址，必须强引用。
            captured_inner = inner
            captured_args = tuple(_weak_ref_if_tensor(a) for a in args)
            captured_kwargs = {k: _weak_ref_if_tensor(v) for k, v in kwargs.items()}
            captured_output = output

            def replay_fn():
                new_out = captured_inner(*captured_args, **captured_kwargs)
                return _copy_output(captured_output, new_out)

            capture.cuda_graph._break_fns.append(replay_fn)
            # 为剩余 forward 开启新捕获段。
            capture._begin_new_segment()
            return output

        return wrapper

    return decorator
```

评论区精华

merrymercy 主导了 review，核心争议有三。一是资格判定重复：他点名 can_run_graph、dp_attn 的 can_run_breakable_cuda_graph 与新引入的 schedule_batch_replay_eligible 三处并存，要求“unify all of these conditions in a single place”；Oasis-Git 随后多轮重构，最终收敛为单一 can_replay_locally 谓词 + 两个薄适配 + dp_attn 只做投票传输（commit 1b16f0f / 018dc6d）。二是代码形态：merrymercy 质疑“why do you need to define it outside? can the eager_on_graph work on class method?”，并引用 no-getattr-defensive 规则否决 scheduler 里的 getattr 防御链；两者都在后续提交解决（分割节点收进 DeepEPMoE 方法、target_worker 显式解包）。三是基础设施必要性：对捕获期 barrier 提出“do we really need this?”，Oasis-Git 以 DeepEP NORMAL 100s CPU 硬超时与段拆卸的 rank 抖动为由保留；对 shared experts 注释“does not make sense”，Oasis-Git 解释双流重叠主要服务 decode，并随后发现并修复了跨捕获段 event 等待导致的 DSV4-Flash token salad。另有 is_extend_in_batch 上下文问题被标记为“应改但涉及 tc_piecewise 原始实现”，属遗留 TODO。

- 统一三处回放资格判定为单一谓词 (design): 多轮重构后收敛为 can_replay_locally 单一谓词（keyword-only 参数、batch 表示无关），can_run_graph 与 dp_attn 投票均为薄适配；投票适配器方法被内联，dp_attn 只做投票传输。
- eager_on_graph 是否支持类方法 / 分割节点为何定义在类外 (design): 分割节点、capture stub 与装饰后的 a2a_forward_with_output 全部收进 DeepEPMoE 类内成为类方法。
- scheduler 中 getattr 防御链违反仓库规范 (style): 删除 getattr 链，改为 dp_attn 中显式的 make_local_breakable_eligible_fn 并以 BaseSpecWorker.target_worker 显式解包，同时修复了 spec worker 投票静默 permissive 的缺口。
- 捕获期 per-break barrier 是否必要 (question): 保留 barrier_fn：段拆卸（capture_end + instantiate）是慢且 rank 间可变的一步，DeepEP NORMAL dispatch 的 100s CPU 超时是编译期常量无法加长。
- deepseek_v2 shared experts alt-stream 注释与行为 (correctness): BCG 下 alt-stream 重叠禁用，但改为段内 wait_event 汇合 + record_stream 标记，修复 16-token bucket 输出损坏。
- is_extend_in_batch 为何不在 prefill runner 直接置 True (question): 现状保留：break body 内临时置 True 是兼容 tc_piecewise 原始路径的折中，评论者与作者均认同应后续调整，属遗留 TODO。
- 函数内 import 清理 (style): 两个文件的 break-node 相关 import 与 is_deepseek_dsa 均提升到模块级，并验证无循环依赖。

风险与影响

- 风险：
  1. DP 一致性：调度期投票与 forward 时点判定共用 can_replay_locally，但两个时点喂入的字段表示不同（ScheduleBatch.prefix_lens vs ForwardBatch.extend_prefix_lens_cpu），capture_hidden_mode=None 依赖 rank-uniform 假设，任何字段语义漂移仍可能造成 rank 分裂；这是核心路径上最需要回归关注的点。
  2. DeepEP 捕获稳定性：非 8 倍数 bucket 会确定性挂死 a2a 捕获（根因未定位），当前靠 _apply_deepep_adjustments 对齐规避；显式用户 bucket 列表也会被静默改写并打日志，可能违背用户预期。
  3. 内存预留是经验值：commit 历史显示 8GB -> 6.5GB -> 1.5GB+1GB 的多轮实测校准，GLM-5.2 类模型 auto mem-fraction 基线缺陷（无 BCG 也 OOM）被有意排除在本 PR 之外，显式 mem-fraction 配置下仍可能 OOM。
  4. 功能回退：TBO 与 BCG 组合、非 DeepEP a2a（pplx）与 BCG 组合均被禁用；DSA 前缀豁免依赖 set_dsa_prefill_impl 强制 use_mha=False 的不变量，若该埋点被绕过会直接破坏捕获状态。
  5. 测试覆盖单薄：13 个文件仅 1 行测试改动，核心逻辑靠 test_dsa_glm52_nvfp4_dp_mtp.py 等 B200 端到端用例兜底，CI 覆盖不到的组合（如 EAGLE + MTP + deepep）回归风险偏高。 - 影响：影响用户与系统：DeepSeek 系（GLM-5.2、DeepSeek-V3.2、DSV4-Flash）在 DSA + DeepEP a2a 配置下默认获得 breakable prefill CUDA graph，prefill 端到端显著加速（基准对比 105.9s -> 48.7s）

；所有 a2a MoE 模型因分割节点上移到 DeepEPMoE.forward 而受益。团队影响：调度器 DP 共识结构变化影响所有 dp 注意力部署；speculative worker 的 model_runner 注入修复了 MTP/EAGLE 下投票静默 permissive 的潜在正确性缺口。影响程度为中等偏大——默认开启但仅限 DeepEP a2a 后端，非 DeepEP 配置行为不变；BCG 捕获时间（56-105s）与 1 GB 额外显存预留是部署者需要知晓的代价。 - 风险标记：核心路径变更，DP 一致性敏感，测试覆盖单薄，捕获稳定性依赖规避，内存预留经验值

关联脉络

- PR #33013 config: read resolved config via namespace accessors: 与 31987 同改 scheduler.py、eagle_worker_v2.py 等文件，且 31987 把 resolved_view 提升为 server_args 模块级 import 并新增 resolved_view(self) 调用点，两者在配置解析路径上相互依赖。
- PR #33012 runtime_context: record the publishing process role: 同样改 scheduler.py 与 speculative worker 链路，角色化 publish 与 31987 中 BaseSpecWorker.target_worker 解包、dp_attn adapter 接线改动处于同一调度器初始化区域。
- PR #32690 [Fix] missing max_context_len on HybridAttnBackend: 同属 attention backend + speculative decoding 验证路径的正确性修复，与 31987 的 BCG 捕获元数据与回放判定同处一个子系统，后续 regression 可能相互触发。