

PR #56512 完整报告

vllm-project/vllm

[DS V4.1][Engram] Support async prefetch for offloaded engram lookups and engram DP sharding

合并时间: 2026-09-13 17:44

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56512>

执行摘要

- 一句话: Engram 表支持 DP 分片、异步预取与 /dev/shm 共享
- 推荐动作: 值得精读。该 PR 是 DeepSeek V4.1 在 vLLM main 上落地的性能与内存优化关键一环, 三个设计点尤其值得学习: TP-major 的 DP head 分片排序 (保证 gather 后 head 顺序正确)、基于 /dev/shm + cudaHostRegister 的跨进程共享 UVA 表 (含失败回滚与 weakref 生命周期管理)、以及 side-stream 预取与 breakable CUDA graph 的配合 (eager break + 持久 staging buffer)。合并前应重点核对 PP 边界场景是否已修复。

功能与动机

原 PR #56230 的目标是让 DP 副本分片共享同一张 engram 嵌入表, 而不是每副本各持一份全量复制 (表可达约 100 GB/ 层); #56357 进一步提出两个 I/O 优化: 把 offload 查找移到 side stream 异步预取, 以及利用 engram hash 的确定性让同节点 DP 副本通过 /dev/shm 共享同一份 TP 切片。PR body 的 benchmark 显示, 同步查找占用的 host-link 时间落在关键路径上 (8k tokens 约 1.2 ms/ 层), 预取可完全恢复这部分时间; 共享模式则同时省去两个 per-step all_gather 与 rows-select 内核。

实现拆解

1. 配置契约与默认值: vllm/config/engram.py 新增 dp_shared_memory 字段 (默认 False), 通过 _validate_shared_memory model_validator 要求 cpu_offload=True, verify_parallel_config 要求 data_parallel_size > 1 且禁用 elastic EP, verify_load_config 限制 load_format 为 auto/safetensors/pt; 抽出 model_has_engram_layers 供分布式层复用。vllm/config/vllm.py 的 _resolve_and_verify_engram_config 现在对 CUDA + 含 engram 层的模型默认物化 EngramConfig (cpu_offload 默认开启), 并拒绝 DBO/microbatching; vllm/envs.py 移除 legacy 环境变量的警告与显式分支。
2. 节点内 EDP 组: vllm/distributed/parallel_state.py 新增 _ENGRAM_DP 组及 get_engram_dp_group/get_engram_dp_size, _engram_dp_shard_size 用 gcd 与节点连续性检查计算分片大小, initialize_model_parallel 中创建 edp 组并接入 destroy 流程; DeepSeek V4.1 且非 elastic EP 时才创建。
3. NVIDIA 专属实现: 新增 vllm/models/deepseek_v4_1/nvidia/engram.py, 实现 engram_head_shard_rank (TP-major 排序)、engram_gathered_num_tokens (从 DP

metadata 推导 token 槽)、gather_engram_hashes (pad + all_gather)、DPSharedEngramStorage (/dev/shm 文件创建、MAP_SHARED 映射、cudaHostRegister、leader-only 写权重、失败回滚与 weakref 反注册) 以及扩展的 ParallelEngramEmbedding (_get_shard_info 覆盖 DP 分片, _allocate_weights/_storage 分支选择共享存储或 pinned host)。

4. 异步预取与公共层改造: vllm/models/deepseek_v4_1/common/engram.py 保留 hash/gating/TP lookup 等公共逻辑, 新增 dummy_hashes (参与 DP collective 但不污染 hash 历史)、_engram_select_rows/_engram_select_rows_kernel (从 rank-major gathered 缓冲取 token 窗口), 并把 _get_shard_info/_allocate_weights/_storage 拆成可覆写钩子; nvidia/model.py 在 forward 入口对非共享且 EDP>1 的情况先做 gather_engram_hashes, 随后 prepare_embeddings 在 side stream 以半格 grid 启动 lookup, embed() 才等待流, eager_break_during_capture 保证不入 captured graph。
5. 测试与 CI 配套: 新增 tests/distributed/test_engram_dp_shard.py (776 行, 覆盖 TP1xDP4/TP2xDP2、失败注入回滚、共享存储生命周期); 扩充 tests/kernels/test_engram.py (dummy hash、空 head 分片拒绝、JIT 变体复用、预取负控制); 扩充 tests/test_config.py 与 tests/engine/test_arg_utils.py 的配置与 CLI 校验; 并将分布式测试接入 .buildkite/test_areas/distributed.yaml 的 4-GPU job。

关键文件:

- vllm/models/deepseek_v4_1/nvidia/engram.py (模块 核心模块; 类别 source; 类型 core-logic; 符号 engram_head_shard_rank, engram_gathered_num_tokens, gather_engram_hashes, DPSharedEngramStorage): PR 核心新增文件: EDP head 分片、DP hash gather、/dev/shm 共享存储与异步预取全部在这里落地
- vllm/models/deepseek_v4_1/common/engram.py (模块 公共模块; 类别 source; 类型 refactor; 符号 dummy_hashes, _storage, _get_shard_info, _allocate_weights): 公共 Engram 基类被拆分为可覆写钩子, 并新增 dummy_hashes 与 rank-major 选行内核, 支撑 NVIDIA 与 AMD 两条路径
- tests/distributed/test_engram_dp_shard.py (模块 分布式测试; 类别 test; 类型 test-coverage; 符号 _token_counts, _full_table, _make_ids, _reference): 新增 776 行分布式测试, 覆盖 DP 分片重建全表、共享存储失败回滚与生命周期, 是两条主特性的主要验证
- vllm/distributed/parallel_state.py (模块 并行组; 类别 source; 类型 core-logic; 符号 get_engram_dp_group, get_engram_dp_size, _engram_dp_shard_size): 新增节点内 EDP 组创建逻辑与 shard size 计算, 是 DP 分片与共享表的基础设施
- vllm/config/engram.py (模块 配置校验; 类别 source; 类型 configuration; 符号 model_has_engram_layers, _validate_shared_memory, verify_load_config): dp_shared_memory 配置字段与全套校验 (cpu_offload/DP 大小 /load_format/elastic EP) 在这里定义
- tests/kernels/test_engram.py (模块 内核测试; 类别 test; 类型 test-coverage; 符号 test_engram_dummy_hashes_leave_history_untouched, test_engram_rejects_empty_head_shards, test_engram_lookup_reuses_jit_across_token_shapes, test_engram_constructor_honors_offload): 单卡内核测试覆盖 dummy

hash、空 head 分片拒绝、JIT 变体复用与 side-stream 预取负控制

- tests/test_config.py (模块 配置测试; 类别 test; 类型 test-coverage; 符号 test_engram_rejects_elastic_cross_dp, test_engram_dp_shared_memory_requires_cpu_offload, test_engram_dp_shared_memory_config_validation, test_engram_config_defaults_to_none) : 配置契约测试: dp_shared_memory 依赖校验、默认 offload 行为、DBO/microbatching 拒绝
- vllm/models/deepseek_v4_1/nvidia/model.py (模块 模型入口; 类别 source; 类型 core-logic) : DeepSeek V4.1 前向入口接入 gather_engram_hashes 与 dummy 参与逻辑, 是运行时关键路径的接线点
- vllm/config/vllm.py (模块 配置解析; 类别 source; 类型 configuration) : 配置解析与运行时拒绝逻辑: 默认物化 EngramConfig、拒绝 DBO/microbatching
- tests/engine/test_arg_utils.py (模块 参数测试; 类别 test; 类型 test-coverage; 符号 test_engram_config_cli, test_engram_config_cli_optional) : CLI 参数测试补充 dp_shared_memory 的 JSON/ 点号写法与默认 offload 断言
- vllm/envs.py (模块 环境变量; 类别 source; 类型 configuration) : 移除 legacy VLLM_PLE_CPU_OFFLOAD 的警告逻辑, 默认开启 CPU offload 是行为变更的一部分
- .buildkite/test_areas/distributed.yaml (模块 CI 配置; 类别 config; 类型 configuration) : 把新分布式测试接入 4-GPU CI job, 解决 review 指出的零覆盖 blocker

关键符号: engram_head_shard_rank, engram_gathered_num_tokens, gather_engram_hashes, DPSharedEngramStorage._allocate, DPSharedEngramStorage.load_weight, ParallelEngramEmbedding._get_shard_info, ParallelEngramEmbedding._allocate_weights, ParallelEngramEmbedding._storage, ParallelEngramEmbedding.forward, NgramHashState.dummy_hashes, _engram_select_rows, _engram_dp_shard_size, get_engram_dp_group, EngramConfig._validate_shared_memory, VllmConfig._resolve_and_verify_engram_config

关键源码片段

vllm/models/deepseek_v4_1/nvidia/engram.py

PR 核心新增文件: EDP head 分片、DP hash gather、/dev/shm 共享存储与异步预取全部在这里落地

```
def _allocate(self, num_bytes: int) -> torch.Tensor:
    """Map and register one physical allocation across a node-local DP group."""
    from vllm.distributed.device_communicators.shm_broadcast import (
        check_shm_free_space,
    )

    group = self.group
    with ExitStack() as stack:
        path, error = None, None
        if group.rank_in_group == 0:
            # 只有 EDP rank 0 创建 /dev/shm 后备文件, 其余副本通过 broadcast 拿到路径
            try:
```

```

        check_shm_free_space(num_bytes)
        backing_file = stack.enter_context(
            tempfile.NamedTemporaryFile(
                prefix="vllm_engram_", dir="/dev/shm"
            )
        )
        backing_file.truncate(num_bytes)
        path = backing_file.name
    except Exception as exc:
        error = f"{type(exc).__name__}: {exc}"
    path, error = group.broadcast_object((path, error))
    if error is not None:
        raise RuntimeError(
            "Engram shared-memory creation failed on EDP rank 0: " + error
        )

mapping = owner = tensor = finalizer = None
stage = "open"
try:
    try:
        with open(path, "r+b") as file:
            stage = "mmap"
            mapping = mmap.mmap(file.fileno(), num_bytes, flags=mmap.MAP_SHARED)
            stage = "cudaHostRegister"
            owner = np.frombuffer(mapping, dtype=np.uint8)
            pointer = owner.ctypes.data
            tensor = torch.from_numpy(owner)
            result = torch.cuda.cudart().cudaHostRegister(pointer, num_bytes, 0)
            if result.value != 0:
                raise RuntimeError(f"cudaHostRegister failed: {result}")
            finalizer = weakref.finalize(owner, self._unregister, mapping, pointer)
            finalizer.atexit = False # type: ignore[misc]
            # UVA 帮助函数否则会分配一份私有 pinned 拷贝，必须确认 CUDA 识别共享注册
            if not tensor.is_pinned():
                raise RuntimeError(
                    "CUDA did not recognize the shared Engram registration"
                )
    except Exception as exc:
        error = f"{stage}: {type(exc).__name__}: {exc}"

    errors: list[str | None] = [None] * group.world_size
    # 在 leader unlink 文件前同步所有副本的映射 / 注册结果，兼作内存栅栏
    torch.distributed.all_gather_object(errors, error, group=group.cpu_group)
    failures = "; ".join(
        f"EDP rank {rank}: {error}"
        for rank, error in enumerate(errors)
        if error is not None
    )
    if failures:

```

```

        raise RuntimeError(
            "Engram shared-memory initialization failed: " + failures
        )
    assert tensor is not None
    return tensor
except Exception:
    if finalizer is not None:
        finalizer()
    tensor = owner = None
    if mapping is not None:
        mapping.close()
    raise

```

vllm/distributed/parallel_state.py

新增节点内 EDP 组创建逻辑与 shard size 计算，是 DP 分片与共享表的基础设施

```

def _engram_dp_shard_size(
    world_size: int,
    data_parallel_size: int,
    replica_size: int,
) -> int:
    """Size of DP shards aligned with complete replicas and node boundaries."""
    node_count = get_node_count()
    replicas_per_node, remainder = divmod(world_size, node_count * replica_size)
    if remainder:
        return 1
    # 分片大小必须同时整除 DP 组大小和每节点副本数
    shard_size = gcd(data_parallel_size, replicas_per_node)
    if shard_size == 1 or node_count == 1:
        return shard_size
    ranks_per_node = replicas_per_node * replica_size
    for start in range(0, world_size, ranks_per_node):
        # 多节点场景要求每个节点内的 rank 连续，否则无法构成节点内 EDP 组
        same_node = in_the_same_node_as(get_world_group().cpu_group, start)
        node_ranks = [rank for rank, local in enumerate(same_node) if local]
        if node_ranks != list(range(start, start + ranks_per_node)):
            return 1
    return shard_size

```

评论区精华

ywang96 的 CHANGES_REQUESTED 提出两个 blocker：其一，[tests/distributed/test_engram_dp_shard.py](#) 未接入任何 Buildkite job，DP 分片与共享表两个核心特性零 CI 覆盖；其二，[deepseek_v4_1/nvidia/model.py](#) 的 `elif get_engram_dp_size() > 1:` 分支在 PP > 1 时对 `self.engram_hash/input_ids` 未做 None 再检查，无 engram 层的 stage 会 `AttributeError`，且 PP 边界恰好落在 engram 层时该 rank 会 fork side stream 但永不 `embed()`，graph capture 在 `capture_end` 会遇到未 join 的流。ZJY0516 则围绕设计给出建议：把构造函数里的 `assert` 移到 config model_validator、为

cross-DP 共享内存建独立类、将 CUDA 专属代码移入 nvidia 目录、参数名改用 `dp_shared_memory`，并询问 DBO 支持情况；作者逐一回复 done/moved/created a new class，并明确 DBO 与 microbatching 不支持。

- 分布式测试未接入 CI (testing): 已在 .buildkite/test_areas/distributed.yaml 增加 4-GPU job 入口，作者回复 done。
- PP 下 engram_hash 为 None 的解引用崩溃 (correctness): PR 最终合并并通过 CI，但评论中未回放修复细节，合并代码仍可见该分支直接调用 self.egram_hash.dummy_hashes，需在后续使用中验证 PP 场景。
- PP 边界落在 engram 层时 side stream 未 join (correctness): 列为合并前需处理的 blocker；最终状态未在公开评论中确认。
- 共享存储抽象与参数命名 (design): 新增 DPSharedEngramStorage 类，参数统一为 dp_shared_memory，与 EngramConfig 字段对齐。
- 校验位置与加载格式限制 (design): EngramConfig 增加 model_validator, verify_load_config 允许 auto/safetensors/pt（含多线程加载）。
- CUDA 专属代码位置与 DBO 支持 (design): 共享与分片逻辑集中到 vllm/models/deepseek_v4_1/nvidia/engram.py；配置层在 DBO 或 microbatching 时抛 ValueError。

风险与影响

- 风险：
 1. PP 兼容性: review 明确指出 nvidia/model.py 的 EDP 分支对 self.egram_hash 与 input_ids 缺少 None 再检查，PP>1 且无 engram 层的 stage 可能 AttributeError；若 PP stage 边界落在 engram 层，侧流未被 join 会在 breakable graph capture 时报错。合并版本未在评论中回放修复细节，属于待确认风险。
 2. 默认行为变更: cpu_offload 默认从 False 变为 True（环境变量 VLLM_PLE_CPU_OFFLOAD 默认启用），现有 Qwen4Exp/DeepSeek V4.1 用户会静默切换到 pinned host + UVA 路径，HBM 占用下降但依赖 UVA 支持。
 3. 共享内存依赖: dp_shared_memory 依赖 /dev/shm 容量（每节点约 94 GB）与共享 IPC namespace，容量不足或容器隔离会导致初始化失败；失败回滚路径虽有测试但真实集群环境未验证。
 4. EDP 组拓扑假设: _engram_dp_shard_size 假设节点内 rank 连续且节点均匀，非均匀拓扑会退化为不创建组，但该退化路径的运行时报错有限。
 5. 平台限制: 预取与共享表为 NVIDIA 专属实现，ROCm 仍走 common resident 路径，且 PR body 说明 ROCm 硬件验证未运行。- 影响: 对使用 DeepSeek V4.1 与 Qwen4Exp 的用户是行为级变更: 默认 CPU offload 减少显存占用，TP4xDP8 跨节点场景下每节点只保留一份完整表（约 94 GB），DP 副本间 zero engram 通信; prefetch 对长 prompt 的 TTFT 有 0.4%-0.9% 提升且 decode 无回归。对团队而言，新增 EDP 分布式组与共享存储生命周期管理是后续 Engram/ 多模态嵌入优化的基础，也增加了 distributed 与 config 层的维护面。CI 需要 4-GPU 分布式资源来跑新增测试。- 风险标记: PP 边界场景待确认，默认 CPU offload 行为变更，依赖 /dev/shm 容量与 IPC，

关联脉络

- PR #56230 [Model][EDP][Engram] Support DP sharding for Engram tables: 本 PR 吸收的前身: engram DP head 分片的实现来源, 测试计划与 GSM8K 评测均沿用
- PR #56357 [Perf][Engram] Deepseek V41 Prefetch offloaded engram lookups; share host tables via mmap: 本 PR 吸收的前身: 异步预取与 /dev/shm 共享表的原始提交来源, 因 staging 分支删除自动关闭
- PR #56682 [Bugfix] Avoid repeated dummy initialization and random CPU Engram fills: 同改 vllm/models/deepseek_v4_1/common/engram.py, 与本 PR 的 dummy 初始化与常量权重填充逻辑直接联动
- PR #56349 [ROCm] Auto-enable breakable CUDA graphs for DeepseekV41ForCausalLM: 本 PR 的 side-stream 预取依赖 eager_break_during_capture 与 breakable CUDA graph 机制, 二者在同一模型上相互耦合