

PR #56145 完整报告

vllm-project/vllm

[Core] MRV2 support for fast-prefill

合并时间: 2026-09-11 01:36

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56145>

执行摘要

- 一句话: MRv2 接入 KV-sharing fast prefill 并解除 V2 门禁
- 推荐动作: 建议精读, 重点看三处: 一是 `vllm/v1/worker/gpu/attn_utils.py` 中 `FastPrefillHelper.prepare` 的三条门控 (`not has_prefill / CUDAGraphMode.FULL / num_ubatches > 1`) 与 `buf[num_logits:] = logits_indices[-1]` 的填充手法——这是「缩减 batch 复用到已捕获 cudagraph」的通用范式, 可迁移到其他需要动态缩小 batch 的优化上; 二是 `get_kv_sharing_fast_prefill_eligible_layers` 的「反向扫描取连续后缀 + 跳过但不打断 draft 层」写法, 它把「层注册顺序即执行顺序」这一隐式约定显式化为可测试逻辑; 三是 `make_kv_sharing_fast_prefill_common_attn_metadata` 从 `assert` 改为回退, 理解 V2 dummy batch (cudagraph 捕获) 为什么必须走完整 batch。评审角度建议关注: 新增的静默降级分支是否需要日志 / 告警, `num_logits_indices` 裁剪契约是否需要断言或文档化, 以及为 `prepare` 的门控与 PCP 分支补单测。

功能与动机

`kv_sharing_fast_prefill` 是针对 YOCO / Gemma3n 这类 KV-sharing 模型的 prefill 优化: 只有真正产生 KV cache 的尾部层需要参与 prefill, 其余共享 KV 的层只需处理 logits 位置, 从而提前退出、节省长 prompt 的 prefill 计算; 而它此前在 V2 model runner 上被显式禁用——`vllm/config/vllm.py` 的 `_get_v2_model_runner_unsupported_features()` 直接把 "KV sharing fast prefill" 加入 unsupported 列表, 注释写着 "Will be added by <https://github.com/vllm-project/vllm/pull/35045>", `CacheConfig.kv_sharing_fast_prefill` 的 docstring 也标注 "KV cache sharing is not supported for MRv2 (v2 model runner)"。本 PR 正是兑现该承诺: 复用 V1 已有的 `create_fast_prefill_custom_backend / make_kv_sharing_fast_prefill_common_attn_metadata` 机制, 把它接进 V2 的 `ModelRunner.prepare_inputs` 与 `build_attn_metadata`, 属于 MRv2 特性对齐 (feature parity) 工作的一环。PR body 为空, 动机主要来自代码注释与配置门禁的移除。

实现拆解

1. 解除 V2 能力门禁: `vllm/config/vllm.py` 删除 `_get_v2_model_runner_unsupported_features()` 中 "KV sharing fast prefill" 的 unsupported 条目 (连同指向 PR#35045 的待办注释); `vllm/config/cache.py` 删除 `kv_sharing_fast_prefill` docstring 里 "KV cache sharing is not supported for MRv2 (v2 model runner)" 的 NOTE。这两处是 V2 对该特性的「能力声明」, 删掉后配置不再被拒绝, 后续执行正确性完全依赖运行期实现。

2. 初始化期识别合格层并改写注意力后端: `vllm/v1/worker/gpu/attn_utils.py` 新增 `get_kv_sharing_fast_prefill_eligible_layers(vllm_config, draft_layer_names)`, 利用「层按执行顺序注册」的约定, 从末尾反向扫描 `Attention` 层, 取 KV-sharing 层的连续后缀; 扫描时遇到 `draft` 层只跳过、不中断 (草稿层注册在目标模型之后, 既不计入也不能截断目标的尾部区间)。 `init_attn_backend` 新增 `draft_layer_names` 参数, 对合格层调用 `create_fast_prefill_custom_backend("FastPrefill", attn_backend)` 包装, 使该层的 `metadata builder` 会把完整 batch 改写成 logits-only batch——这是复用 V1 机制、而非重写 V2 专属路径的关键设计。
3. 逐步决策与元数据准备: 新增 `FastPrefillBatchMetadata (logits_indices_padded / num_logits_indices / max_logits_per_req)` 与 `FastPrefillHelper` (预分配 `logits_indices_buf`、`prepare()` 门控、`cudagraph_manager.dispatch()` 取 `piecewise cudagraph` 尺寸、`buf[num_logits:]` 用最后一个有效索引广播填充以避免残留脏值)。 `model_runner.py` 在 `initialize_kv_cache()` 中按 `self.cache_config.kv_sharing_fast_prefill` 和 `self.pcp_manager is None` 构造 helper, 在 `prepare_inputs()` 中调用 `prepare(logits_indices, num_reqs, cu_num_logits_np, batch_req_state.has_prefill, batch_desc)`, 并在 `shutdown()` 中置空释放; 顺带把 `initialize_mamba_ssu_backend` 的 `use_replayssm` 从 `self.vllm_config.cache_config` 改为 `self.cache_config`。
4. 元数据透传链路与运行期回退: `vllm/v1/worker/gpu/input_batch.py` 给 `InputBatch` 新增 `fast_prefill` 字段 (dummy batch 保持 `None`, 即捕获阶段按完整 batch 跑); `model_states/default.py` 的 `prepare_attn` 把 `input_batch.fast_prefill` 透传给 `build_attn_metadata`; `attn_utils.build_attn_metadata` 新增 `fast_prefill` 参数, 把三个字段写入 `common_attn_metadata_extra_kwargs`; `ubatch_utils.slice_input_batch` 显式置 `fast_prefill=None` 保证 `ubatch` 切片不回落到错误语义。最关键的是 `vllm/v1/attention/backends/utils.py: make_kv_sharing_fast_prefill_common_attn_metadata` 原来对 `logits_indices_padded / num_logits_indices` 直接 `assert`, 现改为 `None` 时原样返回 (覆盖 `cudagraph` 捕获、纯 `decode` 等「本步未启用」场景), 否则 V2 的 dummy batch 会直接打断捕获流程。
5. 测试与 CI 配套: `tests/v1/worker/test_attn_utils.py` 新增 `test_get_kv_sharing_fast_prefill_eligible_layers`, 用 `monkeypatch` 伪造 `get_layers_from_vllm_config / get_shared_kv_cache_layers`, 覆盖无共享、YOCO 式尾部连续区间、区间被非共享层打断、`draft` 层收录与排除、开关关闭 5 类情形; `tests/v1/e2e/general/test_kv_sharing_fast_prefill.py` 新增 `use_v2_model_runner` 维度 (`(False,False) / (True,False) / (True,True)`, 有意省略 `(False,True)` 以省去约 9 分钟 `torch.compile` 时间), 并新增 `assert llm.llm_engine.vllm_config.use_v2_model_runner == use_v2_model_runner` 防止静默回退到 V1; `tests/v1/worker/test_gpu_model_runner_v2.py (+1/-1)` 与 `.buildkite/test_areas/engine.yaml (+1/-1)` 做了小幅同步, 材料未给出具体内容。

关键文件:

- `vllm/v1/worker/gpu/attn_utils.py` (模块 注意力元数据; 类别 `source`; 类型 `core-logic`; 符号 `FastPrefillBatchMetadata, FastPrefillHelper, init, prepare`): 核心实现文件: 新增 `FastPrefillBatchMetadata, FastPrefillHelper` (`prepare` 门控 + `cudagraph` 补齐 + 共享 `buffer`), 新增 `get_kv_sharing_fast_prefill_eligible_layers, init_attn_backend` 接收

draft_layer_names 并对合格层套 create_fast_prefill_custom_backend, build_attn_metadata 新增 fast_prefill 参数注入 metadata kwargs。

- vllm/v1/worker/gpu/model_runner.py (模块 模型运行器; 类别 source; 类型 data-contract; 符号 prepare_inputs, initialize_kv_cache, shutdown) : 把 helper 接入 V2 主循环: initialize_kv_cache 中按开关与 PCP 条件构造 FastPrefillHelper, prepare_inputs 中调用 prepare 并写入 InputBatch, shutdown 中释放; 同时把 draft_attn_layer_names 传给 init_attn_backend, 并顺带把 use_replayssm 改为读 self.cache_config。
- vllm/v1/attention/backends/utils.py (模块 注意力后端; 类别 source; 类型 core-logic; 符号 make_kv_sharing_fast_prefill_common_attn_metadata) : 把 make_kv_sharing_fast_prefill_common_attn_metadata 里对 logits_indices_padded / num_logits_indices 的两条 assert 改成 None 检查后原样返回, 为 cudagraph 捕获 (dummy batch) 与未启用 fast prefill 的步骤提供回退, 是整个改动能安全落地到 V2 的前提。
- vllm/config/vllm.py (模块 运行器能力; 类别 source; 类型 configuration; 符号 _get_v2_model_runner_unsupported_features) : 删除 _get_v2_model_runner_unsupported_features() 中 "KV sharing fast prefill" 的 unsupported 条目 (含指向 PR#35045 的待办注释), 使 V2 配置不再被拒绝, 是本次能力解锁的入口。
- tests/v1/e2e/general/test_kv_sharing_fast_prefill.py (模块 端到端测试; 类别 test; 类型 test-coverage; 符号 test_kv_sharing_fast_prefill) : e2e 用例新增 use_v2_model_runner 维度并新增 assert llm.llm_engine.vllm_config.use_v2_model_runner == use_v2_model_runner, 显式防止静默回退; 同时有意省略 (False, True) 组合以节省约 9 分钟 CI 时间。
- tests/v1/worker/test_attn_utils.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 test_get_kv_sharing_fast_prefill_eligible_layers, check) : 新增 test_get_kv_sharing_fast_prefill_eligible_layers, 用 monkeypatch 覆盖无共享、YOCO 尾部区间、区间被打断、draft 层收录 / 排除、开关关闭 5 类场景, 是合格层识别逻辑的唯一直接测试。
- vllm/v1/worker/gpu/input_batch.py (模块 输入批次; 类别 source; 类型 dependency-wiring; 符号 InputBatch.fast_prefill) : InputBatch 新增 fast_prefill: FastPrefillBatchMetadata | None = None 字段, 注释明确 dummy (cudagraph 捕获) batch 保持 None 即按完整 batch 执行, 是元数据在 runner 与模型状态之间的载体。
- vllm/config/cache.py (模块 缓存配置; 类别 source; 类型 documentation; 符号 CacheConfig.kv_sharing_fast_prefill) : 删除 kv_sharing_fast_prefill docstring 中 "KV cache sharing is not supported for MRv2 (v2 model runner)" 的说明, 使文档与新的能力声明一致。
- vllm/v1/worker/gpu/model_states/default.py (模块 模型状态; 类别 source; 类型 data-contract; 符号 prepare_attn) : prepare_attn 把 input_batch.fast_prefill 透传给 build_attn_metadata, 是元数据从 batch 进入注意力构建的最后一段接线。

- vllm/v1/worker/gpu/ubatch_utils.py (模块 微批切片; 类别 source; 类型 core-logic; 符号 _slice_input_batch) : _slice_input_batch 显式设置 fast_prefill=None, 确保 ubatch 切片不会继承不匹配的 fast-prefill 元数据 (多 ubatch 场景在 prepare 中本就返回 None)。
- .buildkite/test_areas/engine.yaml (模块 测试编排; 类别 config; 类型 configuration) : CI 测试编排小幅调整 (+1/-1), 与新 e2e 参数化用例保持一致; 材料未给出具体内容。
- tests/v1/worker/test_gpu_model_runner_v2.py (模块 运行器测试; 类别 test; 类型 test-coverage) : V2 model runner 单测小幅同步 (+1/-1), 推测与 unsupported 特性列表 / 配置断言变化有关; 材料未给出具体内容。

关键符号: FastPrefillHelper.init, FastPrefillHelper.prepare, get_kv_sharing_fast_prefill_eligible_layers, init_attn_backend, build_attn_metadata, make_kv_sharing_fast_prefill_common_attn_metadata, ModelRunner.initialize_kv_cache, ModelRunner.prepare_inputs, ModelRunner.shutdown, prepare_attn, _slice_input_batch, test_get_kv_sharing_fast_prefill_eligible_layers, test_kv_sharing_fast_prefill

关键源码片段

vllm/v1/worker/gpu/attn_utils.py

核心实现文件: 新增 FastPrefillBatchMetadata、FastPrefillHelper (prepare 门控 + cudagraph 补齐 + 共享 buffer), 新增 get_kv_sharing_fast_prefill_eligible_layers, init_attn_backend 接收 draft_layer_names 并对合格层套 create_fast_prefill_custom_backend, build_attn_metadata 新增 fast_prefill 参数注入 metadata kwargs。

```
@dataclass(frozen=True)
class FastPrefillBatchMetadata:
    """KV-sharing fast prefill 路径的每步输入。"""

    logits_indices_padded: torch.Tensor # 已按 cudagraph 尺寸补齐的 logits 索引
    num_logits_indices: int # 真实 (未补齐) 的 logits 数量
    max_logits_per_req: int # 单请求最大 logits 数, host 侧已知


class FastPrefillHelper:
    """逐 step 决定是否启用 KV-sharing fast prefill, 并准备好所需的 logits 索引。"""

    def __init__(self, cudagraph_manager: "CudaGraphManager", max_num_tokens: int):
        self.max_num_tokens = max_num_tokens
        self.cudagraph_manager = cudagraph_manager
        # 复用一块预分配的 GPU buffer, 避免每步分配; 写入不足时剩余槽位仍需保持有效
        self.logits_indices_buf = torch.zeros(
            max_num_tokens, dtype=torch.int32, device=cudagraph_manager.device
        )

    def prepare(
```

```

self,
logits_indices: torch.Tensor,
num_reqs: int,
cu_num_logits_np: np.ndarray,
has_prefill: bool,
batch_desc: "BatchExecutionDescriptor",
) -> FastPrefillBatchMetadata | None:
    # 三类情况直接放弃本步：纯 decode（无 prefill）、FULL cudagraph（整图无法承载
    # 缩减后的 batch）、ubatching 多 ubatch（每个 ubatch 需要各自独立的 metadata）
    if (
        not has_prefill
        or batch_desc.cg_mode == CUDAGraphMode.FULL
        or batch_desc.num_ubatches > 1
    ):
        return None
    buf = self.logits_indices_buf
    num_logits = logits_indices.shape[0]
    assert num_logits > 0
    buf[:num_logits].copy_(logits_indices)
    # buf[num_logits:] 可能还残留上一步的索引，用最后一个有效索引在 GPU 侧广播填充，
    # 保证补齐区间不会指向非法请求（这些多余 logits 由 num_logits_indices 裁剪丢弃）
    buf[num_logits:] = logits_indices[-1]
    # 让 KV-sharing 层以「仅 logits」的缩减 batch 命中已捕获的 piecewise cudagraph 尺寸；
    # 捕获前或关闭 cudagraph 时，dispatch 会返回未补齐的数量
    desc = self.cudagraph_manager.dispatch(
        num_reqs=num_reqs,
        num_tokens=num_logits,
        uniform_token_count=None,
        num_active_loras=0,
    )
    num_logits_padded = min(desc.num_tokens, self.max_num_tokens)
    return FastPrefillBatchMetadata(
        logits_indices_padded=buf[:num_logits_padded],
        num_logits_indices=num_logits,
        # 单请求最大 logits 数，直接在 host 侧从 cu_num_logits 推出
        max_logits_per_req=int(np.diff(cu_num_logits_np).max()),
    )

```

```

def get_kv_sharing_fast_prefill_eligible_layers(
    vllm_config: VllmConfig, draft_layer_names: set[str] | None = None
) -> set[str]:

```

"""KV-sharing 层中位于末尾的连续区间才有资格走 fast prefill。

在 You Only Cache Once (<https://arxiv.org/abs/2405.05254>) 等 KV sharing 结构中，只有真正产生 KV cache 的尾部层参与 prefill，从而可以提前退出。层按执行顺序注册，因此合格层就是 KV-sharing 层的连续后缀；草稿层注册在目标模型之后，需要显式排除。

"""

```

if not vllm_config.cache_config.kv_sharing_fast_prefill:
    return set()

```

```

shared_kv_cache_layers = get_shared_kv_cache_layers(vllm_config)
if not shared_kv_cache_layers:
    return set()
eligible_layers: set[str] = set()
attn_layers = get_layers_from_vllm_config(vllm_config, Attention)
for layer_name in reversed(attn_layers):
    # 草稿层「跳过但不打断」：既不计入合格集合，也不会截断目标模型的尾部区间
    if draft_layer_names is not None and layer_name in draft_layer_names:
        continue
    if layer_name not in shared_kv_cache_layers:
        break # 遇到非 sharing 层，连续后缀到此结束
    eligible_layers.add(layer_name)
return eligible_layers

# init_attn_backend(): 对合格层换上 FastPrefill 包装后端，其 metadata builder
# 会把完整 batch 改写成 logits-only batch (复用 V1 的 create_fast_prefill_custom_
# backend)
for layer_name in layer_names:
    attn_backend = attn_layers[layer_name].get_attn_backend()
    if layer_name in fast_prefill_eligible_layers:
        attn_backend = create_fast_prefill_custom_backend(
            "FastPrefill", attn_backend
        )

```

vllm/v1/worker/gpu/model_runner.py

把 helper 接入 V2 主循环：initialize_kv_cache 中按开关与 PCP 条件构造 FastPrefillHelper，prepare_inputs 中调用 prepare 并写入 InputBatch，shutdown 中释放；同时把 draft_attn_layer_names 传给 init_attn_backend，并顺带把 use_replayssm 改为读 self.cache_config。

```

# initialize_kv_cache(): 仅在开启 KV-sharing fast prefill 且没有 PCP 时创建 helper。
# PCP (prefill context parallel) 当前不走这条路径，因此直接不启用而不是报错
if self.cache_config.kv_sharing_fast_prefill and self.pcp_manager is None:
    self.fast_prefill = FastPrefillHelper(
        self.cudagraph_manager, self.max_num_tokens
    )

# 把 speculator 的草稿注意力层名交给 init_attn_backend,
# 用于在 get_kv_sharing_fast_prefill_eligible_layers 中被排除，避免草稿层混进合格后缀
draft_attn_layer_names = None
if isinstance(self.speculator, DraftModelSpeculator):
    draft_attn_layer_names = self.speculator.draft_attn_layer_names
self.attn_groups, attn_cg_support, self.kernel_block_sizes = init_attn_backend(
    self.kv_cache_config,
    self.vllm_config,
    self.device,
    draft_layer_names=draft_attn_layer_names,
)

# prepare_inputs(): logits_indices 与 batch_desc 就绪后逐 step 决策,

```

```

# 结果经 InputBatch.fast_prefill 一路透传到 attention metadata 构建
fast_prefill = None
if self.fast_prefill is not None:
    fast_prefill = self.fast_prefill.prepare(
        logits_indices,
        num_reqs,
        cu_num_logits_np,
        batch_req_state.has_prefill,
        batch_desc,
    )

```

vllm/v1/attention/backends/utils.py

把 `make_kv_sharing_fast_prefill_common_attn_metadata` 里对 `logits_indices_padded / num_logits_indices` 的两条 `assert` 改成 `None` 检查后原样返回，为 `cudagraph` 捕获（dummy batch）与未启用 `fast prefill` 的步骤提供回退，是整个改动能安全落地到 V2 的前提。

```

def make_kv_sharing_fast_prefill_common_attn_metadata(
    common_attn_metadata: CommonAttentionMetadata,
) -> CommonAttentionMetadata:
    if common_attn_metadata.max_query_len == 1:
        # 全部请求都是 decode（按 1 个 token 处理），无需走 fast prefill 路径
        return common_attn_metadata

    if (
        common_attn_metadata.logits_indices_padded is None
        or common_attn_metadata.num_logits_indices is None
    ):
        # 本步未启用 fast prefill（例如 cudagraph 捕获阶段，或纯 decode 步骤）：
        # KV-sharing 层按完整 batch 执行，直接返回未修改的 metadata。
        # 这里原来是两条 assert，V2 的 dummy batch 会命中该分支，因此必须优雅回退。
        return common_attn_metadata

    logits_indices_padded = common_attn_metadata.logits_indices_padded
    num_logits_indices = common_attn_metadata.num_logits_indices
    # 去掉 cudagraph 补齐部分，只保留真实索引
    logits_indices = logits_indices_padded[:num_logits_indices]

```

评论区精华

本 PR 的评审讨论非常薄，几乎没有实质技术交锋，可作为「维护者自研 + 直接合并」型改动的样本：

claude[bot] (COMMENTED) : 「This pull request is from a fork — automated review is disabled. A repository maintainer can comment `@claude review` to run a one-time review.」

自动 review 因来自 fork 被跳过，且没有人手动触发 `@claude review`；
`review_comments_count` 为 0，说明没有任何行内讨论。

WoosukKwon (APPROVED) : 「Thanks!」

唯一的人工评审是维护者 WoosukKwon 的一句致谢式批准，未对 `FastPrefillHelper.prepare` 的门控条件（FULL cudagraph / 多 ubatch / 无 prefill）、PCP 下不创建 helper、以及把 `assert` 降级为运行时回退这些设计选择提出异议。

njhill (issue 评论) : 「/ci run」× 2 → Buildkite CI #87996 (commit 9b093b66af0e) 与 #88010 (commit 8e00cb479621)

两次触发对应的 commit 不同，说明 head 至少被重推 /amend 过一次后才重新验证；最终以单 commit 8e00cb479621 合并。

- fork PR 的自动 review 被跳过，缺少实质性代码讨论 (other): 自动评审未执行，人工评审也未展开细节，最终由维护者一句致谢批准合并。
- 维护者以「Thanks!」直接批准，未质疑门控与回退设计 (design): 设计决策完全由作者自定并随代码注释表达，评审侧没有形成交锋或额外约束。
- CI 两轮重跑，head commit 被重推 (testing): 最终以单 commit 8e00cb479621 通过 CI 并合并；仓库侧只记录到 CI 触发记录，没有失败详情。

风险与影响

- 风险：
 1. 静默降级路径较多，正确性依赖回退语义：FastPrefillHelper.prepare 在「纯 decode」「FULL cudagraph」「多 ubatch」三种情况下返回 None，PCP 下更是根本不构造 helper，ubatch_utils.slice_input_batch 也硬编码 fast_prefill=None。这些分支都不会报错或告警，用户会以为开启了 fast prefill 但实际按完整 batch 执行；且 vllm/v1/attention/backends/utils.py 中原有的两条 assert 被替换为原样返回，若未来有新的调用方依赖「必须带 fast-prefill 元数据」的隐含契约，会从「立即失败」变成「静默跑错语义」。
 2. cudagraph 尺寸耦合：prepare() 内直接调用 self.cudagraph_manager.dispatch(num_reqs=num_reqs, num_tokens=num_logits, ...)，用返回的 desc.num_tokens 反推补齐长度，再用 min(desc.num_tokens, self.max_num_tokens) 截断。运行时 dispatch 结果与捕获时注册的 piecewise 尺寸必须严格一致，否则要么捕获不到对应尺寸、要么补齐后仍走不进已捕获的图；这层不变式没有专门测试守护。
 3. 共享 buffer 的填充语义：buf[:num_logits].copy_(logits_indices) 之后把 buf[num_logits:] 全量写成 logits_indices[-1]，这些「伪索引」在补齐区间会产出被丢弃的 logits。任何绕过 num_logits_indices 裁剪的下游消费（例如未来新增的 logits 收集路径）都会读到重复请求的 logits，属于隐藏的隐式契约。
 4. max_logits_per_req 依赖 host 侧数组：int(np.diff(cu_num_logits_np).max()) 直接消费 cu_num_logits_np，若该数组在某条新路径上与 GPU 侧 cu_num_logits 不同步，会传入错误的 per-request 上界；同函数内 assert num_logits > 0 在「有 prefill 但 logits 为空」的边界（如纯 logprobs 场景）可能触发断言。
 5. 测试覆盖不均匀：单测只覆盖了 get_kv_sharing_fast_prefill_eligible_layers 的层选择逻辑，e2e 用例只验证端到端答案与「未静默回退」，FastPrefillHelper.prepare 的三条门控分支、PCP 不创建 helper 的分支均无直接测试；tests/v1/worker/test_gpu_model_runner_v2.py 与 .buildkite/test_areas/engine.yaml 的小改动材料未展开，无法确认是

否覆盖了新行为。

6. 门禁移除后组合矩阵扩大：V2 不再拒绝 kv_sharing_fast_prefill，该开关与 DBO/ubatching、PCP、FULL cudagraph、speculative decoding 的组合都会进入实际执行路径，其中多数组合只能靠上面的静默降级兜底，缺少显式的不支持提示。
- 影响：用户侧：使用 Gemma3n / YOCO 类 KV-sharing 模型并已切到 V2 model runner 的用户，现在可以真正开启 kv_sharing_fast_prefill，长 prompt prefill 只需在尾部 KV 生成层上计算，预期显著降低 prefill 计算量与 TTFT；此前该开关在 V2 上会被配置校验直接拒绝，属于从「不可用」到「可用」的能力补齐。系统侧：改动落在 V2 核心热路径 ModelRunner.prepare_inputs → build_attn_metadata → 自定义 backend 的 metadata builder，但主链路只增加一次 prepare() 调用与一次 cudagraph_manager.dispatch()（均为轻量 host 操作），对不开该特性的用户基本零影响；不支持的组合（FULL cudagraph、多 ubatch、PCP）行为保持原有语义不变。
 - 团队侧：这次合并清掉了 MRv2 特性清单上的一条明确待办（代码注释里的 PR#35045），也说明 MRv2 正在从「新 runner 只能跑子集特性」转向「逐步对齐 V1 全部优化」，后续同源开关（如 mamba cache mode、logits processors）的迁移可复用这次「包装 V1 后端 + V2 侧准备元数据 + 运行期优雅回退」的模式。
 - 风险标记：核心推理路径变更，静默降级无告警，cudagraph 尺寸耦合，prepare 门控缺少单测，配置门禁移除后组合面扩大

关联脉络

- PR #55819 [Perf] Use UVA-backed contents for MRV2 apply_write: 同属 MRv2 (v2 model runner) 优化线，改动同样落在 vllm/v1/worker/gpu/ 下的 buffer/ 热路径代码，说明 V2 runner 正在持续补齐性能与特性能力，本 PR 可视为该脉络上的能力对齐一环。
- PR #54713 [BugFix] Retain both replay boundaries so an EAGLE resend of a block-aligned prompt still hits: 同样涉及 v1 侧 KV 复用的正确性边界（KV cache 命中/replay 边界），与本 PR 依赖的 KV-sharing 语义、logits 索引与 attention metadata 裁剪契约处在相邻的正确性面上，可对照理解缓存复用类改动的回归风险。
- PR #55531 [KV Connector] Support symmetric DCP disagg for hybrid mamba models: 该 PR 与并行拓扑（DCP/PCP）下的 KV cache 组和 attention metadata 构建强相关；本 PR 在 init_attn_backend/model runner 初始化阶段同样以并行配置（pcp_manager is None）作为 fast prefill 的启用条件，属于同一类「并行拓扑 × KV 特性」的交叉领域。