PR #37170 完整报告

sgl-project/sglang

[unified-memory] Drop the vacated 'dense' qualifier and the restating comments

合并时间：2026-08-31 15:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37170

执行摘要

- 一句话：清理 unified-memory 遗留的 dense 术语，统一为 kernel-facing
- 推荐动作：值得快速浏览（约 10-15 分钟）：重点阅读 PR body 的 Deliberately untouched 清单，这是区分「死术语」与「活术语」的决策框架，对大型重构后的术语治理有普适参考价值。不建议逐文件精读；若后续在 unified-memory 上工作，务必记住 dense（保留场景）与 kernel-facing（id 空间）的分界。另建议跟进 Codex 遗留的两个 P1 评论，避免它们随堆叠链静默合入主干。

功能与动机

PR body 明确说明：dense named a per-layer KV view only in contrast to the strided MHA view, which no longer exists, so the word marks nothing. 即 dense 一词原本只用于与已被删除的 strided MHA 视图区分，如今已无对比对象，继续使用会误导读者。方案是：修饰视图时删除该词；命名 id 空间时改为 kernel-facing（翻译器已长期使用的术语）。同时刻意保留仍携带语义的 dense 用法：MoE dense layers、first_k_dense_replace、DSA fallback、minimax dense_layer_ids、dual-chunk、trtllm_mla 的 dense query indptr、mamba conv/SSM 的 dense pitch——这些位置的对比真实存在。第二个动机是注释债务清理：8 份 mock-runner 复述注释、4 份 None 初始化说明、1 份脱离分支的 check the fast path、1 份语法不通的 restatement，以及 em-dash 与 .claude/rules/comment-style.md 的不一致。

实现拆解

变更入口：PR body 自述为 unified-memory 堆叠链遗留的两项清理，无关联 Issue，无新增功能。

1. KV 术语迁移（dense -> 删除 / kernel-facing）：核心原则是区分两种语义——dense 修饰「视图」时删除（strided MHA 视图已不存在，per-layer 视图就是唯一形态）；dense 命名「id 空间」时改为 kernel-facing，与 KVIndexTranslator 既有术语（translate_kv_loc_for_kernel、build_index_table）对齐。具体落点包括：test_unified_mla_block_table.py 的 TestDenseBlockTable -> TestBlockTable、TestFa3MetadataDenseBlockTable -> TestFa3MetadataBlockTable 及多个测试方法；test_unified_mla_views.py 的 TestDenseMLAViews -> TestMLAViews、TestTranslateKvLocDense -> TestTranslateKvLocForKernel、test_dense_matches_formula_ps1 -> test_kernel_id_matches_formula_ps1 等；

test_unified_mha_views.py 的 4 个测试类名；test_multi_ended_allocator.py 的 TestSWACompositeDenseSurface -> TestSWACompositeKernelIdSurface；test_page_major_backend_allowlist.py 的 DENSE_MLA_BACKENDS -> PER_LAYER_VIEW_MLA_BACKENDS（此处的 dense 修饰视图，因此改用 per-layer view 而非 kernel-facing）；test_kv_index_translator.py、test_pd_envelope_transfer_layout.py、test_full_loc_fast_path.py 的测试方法同步更名。

2. 运行时源码注释同步：memory_pool.py 中 kernel_page_blocks 的注释由 only where the per-layer views are dense 改为 only for the unified pool's per-layer views，错误信息 dense KV rows 改为 KV rows；trtllm_mla_backend.py 两处 CUDA graph 捕获注释改为 kernel-facing KV write loc；kv_read_table.py、unified_memory_pool.py 各 1-2 行注释微调。
3. 评论清理：删除 8 份 mock-runner 复述自身调用的注释、4 份解释字段为何初始化为 None 的注释（house rule 已约定）、1 份脱离所描述分支的重复 check the fast path 注释、1 份语法不通的 restatement；该堆叠链引入的 em-dash 统一替换为 --，符合 .claude/rules/comment-style.md。
4. 验证与配套：无新增测试（全部为既有测试重命名），无配置 / schema / 部署改动；标识符文件局部性由作者人工验证，测试类与方法名无外部引用；CI 覆盖 base、extra、AMD ROCm 7.2 三个矩阵。

关键文件：

- test/registered/unit/mem_cache/test_unified_mla_block_table.py（模块 MLA 块表；类别 test；类型 test-coverage；符号 TestBlockTable, TestFa3MetadataBlockTable, test_block_table_matches_reference, test_agrees_with_token_level_translate）：本 PR 中符号重命名最密集的测试文件：TestDenseBlockTable / TestFa3MetadataDenseBlockTable 与 4 个测试方法去 dense 化，并同步更新模块 docstring 中的块表公式（kernel_page = v2p[virtual_page] * layer_num），是理解术语迁移的入口。
- test/registered/unit/server_args/test_page_major_backend_allowlist.py（模块 后端白名单；类别 test；类型 test-coverage；符号 PER_LAYER_VIEW_MLA_BACKENDS, PER_LAYER_VIEW_MHA_BACKENDS, test_per_layer_view_mla_backends_allowed_under_unified_mla, test_per_layer_view_mha_backends_allowed_for_uniform_row_models）：常量 DENSE_MLA_BACKENDS / DENSE_MHA_BACKENDS 改为 PER_LAYER_VIEW_*，测试方法同步改名；白名单语义从 dense 视图转为 per-layer 视图，因为 strided 视图已被移除。
- test/registered/unit/mem_cache/test_unified_mla_views.py（模块 MLA 视图；类别 test；类型 test-coverage；符号 TestMLAViews, TestTranslateKvLocForKernel, test_move_then_readback, test_kernel_id_matches_formula_ps1）：TestDenseMLAViews -> TestMLAViews、TestTranslateKvLocDense -> TestTranslateKvLocForKernel，反映 id 空间命名从 dense 收敛到 kernel-facing。
- test/registered/unit/mem_cache/test_multi_ended_allocator.py（模块 分配器；类别 test；类型 test-coverage；符号 TestSWACompositeKernelIdSurface, test_full_kernel_translate_matches_formula,

test_kernel_translate_accepts_an_int32_page_table,test_swa_kernel_tombstone_still_lands_on_sink）：SWA 复合分配器的测试面从DenseSurface 更名为 KernelIdSurface，涉及 translate_loc_from_full_to_swa 与tombstone 回落到 sink 的测试。

- test/registered/unit/mem_cache/test_unified_mha_views.py（模块 MHA 视图；类别 test；类型 test-coverage；符号 TestMHASpecSurface, TestMHAViews, TestUnifiedKVPoolViews, TestFactoryViews）：MHA 视图测试类三处改名（SpecSurface / MHAViews / UnifiedKVPoolViews / FactoryViews），工厂测试明确 kernel-facing 语义。
- test/registered/unit/mem_cache/test_kv_index_translator.py（模块 索引翻译；类别 test；类型 test-coverage；符号 test_read_table_matches_reference_across_multipliers, test_swa_write_loc_round_trips_from_full_side）：translator 是 kernel-facing 术语的原产地，本 PR 让测试命名（test_read_table_matches_reference_across_multipliers、test_swa_write_loc_round_trips_from_full_side 等）与之一致。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic）：运行时源码中仅剩的 dense 措辞清理：kernel_page_blocks 注释改为 unified pool's per-layer views，错误信息从 dense KV rows 改为 KV rows。
- python/sglang/srt/layers/attention/trtllm_mla_backend.py（模块 MLA 后端；类别 source；类型 core-logic）：CUDA graph 捕获相关注释从 DENSE KV write loc 改为 kernel-facing KV write loc，与翻译器命名对齐。

关键符号：test_block_table_matches_reference, test_agrees_with_token_level_translate, test_per_layer_view_mla_backends_allowed_under_unified_mla, test_per_layer_view_mha_backends_allowed_for_uniform_row_models, test_kernel_id_matches_formula_ps1, test_kernel_id_follows_compaction, test_full_kernel_translate_matches_formula, test_swa_kernel_tombstone_still_lands_on_sink, test_read_table_matches_reference_across_multipliers, test_swa_write_loc_round_trips_from_full_side, test_page_envelope_matches_per_layer_views, test_rebind_emits_kernel_facing_full_and_build_derives_swa

关键源码片段

test/registered/unit/mem_cache/test_unified_mla_block_table.py

本 PR 中符号重命名最密集的测试文件：TestDenseBlockTable / TestFa3MetadataDenseBlockTable 与 4 个测试方法去 dense 化，并同步更新模块 docstring 中的块表公式（kernel_page = v2p[virtual_page] * layer_num），是理解术语迁移的入口。

```
# unified-memory 的 MLA block table 测试：原 TestDenseBlockTable /
TestFa3MetadataDenseBlockTable
# 在本 PR 中去掉 dense 前缀。dense 原本只在与已删除的 strided MHA 视图对比时有意义，
# 现在统一使用 translator 已采纳的 kernel-facing 术语描述 id 空间。

def _reference(req_to_token, req_pool_indices, seq_lens, page_size, *, v2p, mult):
    # Python 参考实现：virtual token -> virtual page -> physical page -> kernel id。
```

```
    # 逐元素推导 kernel-facing 公式的期望值，供 CUDA 侧 build_kv_read_table 的结果对照。
    bs = req_pool_indices.shape[0]
    max_blocks = (int(seq_lens.max().item()) + page_size - 1) // page_size
    ref = torch.full((bs, max_blocks), -1, dtype=torch.int64, device=_DEV)
    for r in range(bs):
        n_pages = (int(seq_lens[r].item()) + page_size - 1) // page_size
        row = req_to_token[int(req_pool_indices[r].item())]
        # 每个 page 取首 token 换算 virtual page，再经 v2p 映射到 physical page
        virt_pages = row[: n_pages * page_size : page_size] // page_size
        pages = v2p[virt_pages.long()] if v2p is not None else virt_pages.long()
        ref[r, :n_pages] = pages * mult # mult = layer_num 时即为 kernel-facing id
    return ref


@unittest.skipUnless(_HAS_CUDA, 'requires CUDA')
class TestBlockTable(unittest.TestCase):
    # 原 TestDenseBlockTable；_make_batch 构造带非恒等 v2p 置换的 ragged batch

    def test_block_table_matches_reference(self):
        # 原 test_dense_block_table_matches_reference：规范构建与 Python 参考逐位一致
        for page_size in (1, 32, 64):
            rt, rpi, sl, v2p = self._make_batch(page_size)
            got = _fill_block_table(rt, rpi, sl, page_size, v2p=v2p, mult=_LAYERS)
            want = _reference(rt, rpi, sl, page_size, v2p=v2p, mult=_LAYERS)
            self.assertTrue(
                torch.equal(got.long(), want),
                f'page_size={page_size}: {got} != {want}',
            )

    def test_agrees_with_token_level_translate(self):
        # 原 test_agrees_with_token_level_dense_translate：flashinfer 按 token 走
        # translate_kv_loc_for_kernel，trtllm 路径在 kernel 内构建 page id，
        # 两条路径必须指向同一个 kernel-facing page 块
        page_size = 64
        rt, rpi, sl, v2p = self._make_batch(page_size)
        block_table = _fill_block_table(rt, rpi, sl, page_size, v2p=v2p, mult=_LAYERS).long()
        for r in range(rt.shape[0]):
            n = int(sl[r].item())
            virt_tokens = rt[r, :n].long()
            # token 级 kernel-facing 公式：page 经 v2p 缩放，块内 offset 保留
            kernel_tokens = (
                v2p[virt_tokens // page_size] * (page_size * _LAYERS)
                + virt_tokens % page_size
            )
            first_of_page = kernel_tokens[::page_size]
            n_pages = (n + page_size - 1) // page_size
            self.assertTrue(
                torch.equal(block_table[r, :n_pages] * page_size, first_of_page),
                f'row {r}: block table and token translate disagree',
```

```
            )
```

test/registered/unit/server_args/test_page_major_backend_allowlist.py

常量 DENSE_MLA_BACKENDS / DENSE_MHA_BACKENDS 改为 PER_LAYER_VIEW_*，测试方法同步改名；白名单语义从 dense 视图转为 per-layer 视图，因为 strided 视图已被移除。

```
class TestPageMajorBackendAllowlist(unittest.TestCase):
    # 常量名从 DENSE_MLA_BACKENDS 改为 PER_LAYER_VIEW_MLA_BACKENDS：
    # 统一内存池暴露的就是 per-layer 视图，dense 一词无对比对象
    PER_LAYER_VIEW_MLA_BACKENDS = (
        'fa3',
        'trtllm_mla',
        'flashinfer',
        'cutedsl_mla',
        'tokenspeed_mla',
        'flashmla',
    )
    # 已接线的 per-layer MHA/SWA 视图后端（uniform-row 模型）
    PER_LAYER_VIEW_MHA_BACKENDS = ('fa3', 'fa4', 'flashinfer', 'trtllm_mha')
    # MLA 系内核不得泄漏进 MHA 分支
    MLA_ONLY_BACKENDS = ('trtllm_mla', 'cutedsl_mla', 'tokenspeed_mla', 'flashmla')
    # 没有任何 kernel-facing id 接线：必须保持拒绝
    UNWIRED_BACKENDS = ('cutlass_mla', 'aiter')

    def test_per_layer_view_mla_backends_allowed_under_unified_mla(self):
        # 原 test_dense_mla_backends_allowed_under_unified_mla
        for backend in self.PER_LAYER_VIEW_MLA_BACKENDS:
            self.assertTrue(
                _accepts(backend, use_mla=True),
                f'{backend} should be allowed with the unified-memory MLA pool',
            )

    def test_per_layer_view_mha_backends_allowed_for_uniform_row_models(self):
        # 原 test_dense_mha_backends_allowed_for_uniform_row_models
        for backend in self.PER_LAYER_VIEW_MHA_BACKENDS:
            self.assertTrue(
                _accepts(backend, use_mla=False),
                f'{backend} should be allowed for a uniform-row MHA model',
            )

    def test_unwired_backends_always_rejected(self):
        for backend in self.UNWIRED_BACKENDS:
            for use_mla in (True, False):
                self.assertFalse(
                    _accepts(backend, use_mla=use_mla),
                    f'{backend} has no kernel-facing-id wiring and must be rejected',
                )
```

test/registered/unit/mem_cache/test_unified_mla_views.py

TestDenseMLAViews -> TestMLAViews、TestTranslateKvLocDense -> TestTranslateKvLocForKernel，反映 id 空间命名从 dense 收敛到 kernel-facing。

```
class TestMLAViews(unittest.TestCase):
    # 原 TestDenseMLAViews：统一内存池的 MLA 视图即 per-layer 紧凑视图

    def test_move_then_readback(self):
        # 原 test_move_then_dense_readback：经视图写入 src 页，再移动 KV 缓存，
        # 验证 dst 页读回相同值——确认 page-envelope 移动按 kernel-facing 空间迁移
        ps = 4
        pool, kv_pool = self._make(ps=ps)
        num_pages = pool.max_slots('full') // ps
        src_page, dst_page = num_pages - 3, 5
        for l in range(_L):
            for s in range(ps):
                kv_pool.kv_buffer[l][_kernel_id(src_page * ps + s, ps, _L)] = float(l * ps + s + 1)
        offsets = torch.arange(ps, dtype=torch.int64)
        kv_pool.move_kv_cache(
            (torch.tensor([dst_page])[:, None] * ps + offsets).reshape(-1),
            (torch.tensor([src_page])[:, None] * ps + offsets).reshape(-1),
        )
        for l in range(_L):
            for s in range(ps):
                got = kv_pool.kv_buffer[l][_kernel_id(dst_page * ps + s, ps, _L)]
                self.assertTrue(torch.all(got == float(l * ps + s + 1)), f'(l={l}, s={s})')


class TestTranslateKvLocForKernel(unittest.TestCase):
    # 原 TestTranslateKvLocDense：方法输出的是 kernel-facing id，命名与实现对齐

    def test_kernel_id_matches_formula_ps1(self):
        # 原 test_dense_matches_formula_ps1：ps = 1 时公式退化为 phys * layer_num
        alloc = self._build(ps=1)
        v = alloc.alloc(8)
        self.assertIsNotNone(v)
        phys = alloc.translate_kv_loc(v)
        kernel = alloc.translate_kv_loc_for_kernel(v)
        self.assertTrue(torch.all(kernel == phys * _L))
```

评论区精华

本 PR 无人工讨论（comments_count=1 为 Codex 摘要贴）。3 条 review 评论全部来自 chatgpt-codex-connector[bot]，其 diff_hunk 指向 kv_cache_hook.py、kv_index_translator.py、forward_batch_info.py，均不在本 PR 的 18 个变更文件内，应归因于该堆叠链上的底层提交（PR body 标注 Stacked on #34613）。这些发现对 unified-memory 功能本身仍有参考价值：

> P1・kv_index_translator.py：FA3/FA4 在 context parallel 下按 pad_delta 加宽 raw page table，但 unified 路径随后用更窄的翻译表替换，padding 后的查询可能越界读取

——需用 CP 调整后的 metadata 上界定宽翻译表，或在替换时保留加宽的 sink 列。

P1・forward_batch_info.py：DSPARK 草稿路径直接构造 ForwardBatch，绕过 rebind_write_loc，写位置仍是 virtual id；Triton/TRTLLM MLA 后端假设所有真实 batch 已是 kernel-facing 写 id，可能导致草稿 KV 写错行。

P2・kv_cache_hook.py：prefill CUDA graph 默认禁用发生在内存设置计算之后，reserve_for_graph_mb 仍按 prefill graph 预留约 1.5 GiB（MLA 模型），压低统一内存池容量。

结论：本 PR 变更面与这三条评论无交集，但两条 P1 涉及的功能正确性问题应在 unified-memory 正式落地前确认解决。

- unified-memory 下 prefill CUDA graph 默认禁用顺序导致显存预留偏高（P2）(performance): 建议在内存 hook 之前应用该默认值，或在修改后端后重算内存设置。无后续对话，状态未解决。
- CP padding 导致翻译后的 read table 宽度不足（P1）(correctness): 建议用 CP 调整后的 metadata 上界定宽翻译表，或在替换时保留加宽的 sink 列。无后续对话，状态未解决。
- DSPARK 直接构造的合成 ForwardBatch 未执行写位置 rebind（P1）(correctness): 建议 DSPARK 构造路径调用 translator，或在共享边界统一转换。无后续对话，状态未解决。

风险与影响

- 风险：
  1. 本 PR 自身风险极低：无行为变更，重命名均为文件局部，测试类 / 方法名无外部引用，CI 三个矩阵覆盖。唯一可见风险是「半迁移」——若仓库内仍有 grep 到的旧名 DenseBlockTable 等残留会产生误导，建议合并后做一次仓库级 grep 复核。
  2. 关联风险（非本 PR 引入）：Codex 指出的两个 P1 问题（CP padding 下翻译表宽度不足、DSPARK 合成 batch 未 rebind）位于堆叠链底层代码，若未修复就随主干发布，会影响 unified-memory + FA3/FA4 context parallel 与 DSPARK 路径的正确性。
  3. 兼容性风险：无。常量与测试名属内部符号，不构成公开 API。 - 影响：对用户与运行时零影响（无行为变更）。对系统而言，统一内存池的术语体系收敛到 translator 已定义 kernel-facing，消除了 dense 与 strided 对比的过时心智模型。对团队而言，18 个文件、12+ 测试文件的命名与注释同步，后续 unified-memory 相关 diff 更易检索和评审，也降低新成员把 kernel-facing id 误称 dense 的概率。影响程度低，主要作用于代码可读性与工程一致性。 - 风险标记：18 文件术语重命名，底层 P1 未解决（CP padding / DSPARK），无行为变更依赖 CI 验证

关联脉络

- PR #35245 refactor(unified-memory): translate the KV write location once, at ForwardBatch construction: 同属 unified-memory 重构线，定义了 KVIndexTranslator 与 kernel-facing 写位置语义；本 PR 的 kernel-facing 术语正是该 PR 引入的，本 PR 还同步改写了其测试命名。

- PR #37164 [mem_cache] Move mamba state and retraction_backup into ReqKvInfo: 同属 mem_cache 模块的重构系列，与本 PR 改动同一批 unified-memory 测试与内存池文件，反映该功能线仍在收尾演进。
- PR #37151 [Unified Cache Linker][3/N]: Add backend-independent linker core: unified memory 方向的另一条并行线（缓存链接器），与 unified-memory 术语体系共享 kernel-facing id 概念。