PR #32466 完整报告

sgl-project/sglang

[AMD] Enable gfx1250 sgl-kernel builds

合并时间：2026-08-07 06:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32466

执行摘要

- 一句话：为 AMD gfx1250 启用 sgl-kernel 构建并修正 wheel 标签
- 推荐动作：值得快速阅读，尤其是关注 AMD 硬件支持与 ROCm 打包流程的人。核心看点：在缺少指令集的架构上用编译期空桩保证构建通过、并通过运行时门控维持正确性；以及通过移除 py_limited_api 来修正非 ABI3 扩展的 wheel 标签。规模很小（+14/-4），不需要大型 review。

功能与动机

PR 描述指出：ROCm sgl-kernel 构建此前拒绝 gfx1250，且 QuickReduce 翻译单元引用了 gfx12 上不存在的 raw-buffer 指令；同时 ROCm wheel 被错误标记为 ABI3，而 common_ops 是 CPython 特有扩展。该 PR 的目标是让 gfx1250 能够编译并安装 sgl-kernel，同时修正打包元数据。

实现拆解

按 3 步拆解：

1. 目标架构白名单与编译参数（python/sglang/kernels/aot/setup_rocm.py）：将 gfx1250 加入 amdgpu_target 白名单并更新警告文案；fp8_macro 沿用 "gfx942 用 FNUZ，其余用 E4M3" 的判定，gfx1250 自动复用 E4M3；topk_dynamic_smem_bytes 保持 "gfx942 用 48 KiB，其余用 128 KiB" 的分支，让 gfx1250 与 gfx95x 共享 128 KiB 大 LDS 预算，满足 TopK 内核共享内存需求。
2. QuickReduce 编译兼容（python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce_base.h）：将依赖 llvm.amdgcn.raw.buffer.load/store 的 buffer_load_dwordx4 / buffer_store_dwordx4 声明用 #if !defined(__gfx1250__) 包裹，并在 gfx1250 分支提供空实现。这样共享的 ROCm 扩展在 gfx1250 上也能编译，但 QuickReduce 在运行时仍维持原架构门控，不启用该内核。
3. Wheel 打包标签修正（setup_rocm.py）：移除 options={"bdist_wheel": {"py_limited_api": "cp39"}}。由于 common_ops 依赖 CPython 专有 API（非 ABI3），移除后 wheel 工具会生成 CPython 版本专用标签（如 cp313-），避免 " 声明 ABI3 但实际不兼容 " 的错误打包。

配套验证：PR 描述说明在 gfx1250 上完成了构建、导入和基础 GPU 算子冒烟测试；setup 元数据验证确认了 gfx1250 目标、E4M3 模式、128 KiB TopK 预算与非 ABI3 扩展元数据。

本次没有新增或修改任何测试文件。

关键文件：

- python/sglang/kernels/aot/setup_rocm.py（模块 构建脚本；类别 source；类型 core-logic）：构建入口：gfx1250 目标白名单、FP8 宏与 TopK 共享内存预算的选择、wheel 标签修正都在这里完成，是本次变更的核心。
- python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce_base.h（模块 快速归约；类别 source；类型 core-logic；符号 buffer_load_dwordx4, buffer_store_dwordx4）：QuickReduce 使用的 raw-buffer 指令在 gfx12 上不存在，这里通过空桩让共享扩展在 gfx1250 上仍能编译，同时维持运行时禁用。

关键符号：buffer_load_dwordx4, buffer_store_dwordx4

关键源码片段

python/sglang/kernels/aot/setup_rocm.py

构建入口：gfx1250 目标白名单、FP8 宏与 TopK 共享内存预算的选择、wheel 标签修正都在这里完成，是本次变更的核心。

```
# 支持的目标列表：gfx942 (MI300/MI325)、gfx950 (MI350) 以及新增的 gfx1250 (RDNA4)
if amdgpu_target not in ["gfx942", "gfx950", "gfx1250"]:
    print(
        f"Warning: Unsupported GPU architecture detected '{amdgpu_target}'. "
        "Expected 'gfx942', 'gfx950', or 'gfx1250'."
    )
    sys.exit(1)

# gfx942 使用 FNUZ FP8 表示，其余目标（gfx950 / gfx1250）统一使用 E4M3
fp8_macro = (
    "-DHIP_FP8_TYPE_FNUZ" if amdgpu_target == "gfx942" else "-DHIP_FP8_TYPE_E4M3"
)

# TopK 动态共享内存预算：
# - gfx942: LDS 约 64 KB / 工作组，动态 smem 不能超过约 48 KB
# - gfx95x / gfx1250: LDS 更大，允许原来的 128 KB 动态 smem
# 这样 gfx1250 能直接复用 gfx95x 的 128 KiB 预算，无需为 TopK 内核单独调参。
topk_dynamic_smem_bytes = 48 * 1024 if amdgpu_target == "gfx942" else 32 * 1024 * 4

# hipcc 编译参数：目标架构、BF16/FP8 开关、TopK smem 预算都从这里下发
hipcc_flags = [
    "-DNDEBUG",
    f"-DOPERATOR_NAMESPACE={operator_namespace}",
    "-O3",
    "-Xcompiler",
    "-fPIC",
    "-std=c++17",
    f"--amdgpu-target={amdgpu_target}",
    "-DENABLE_BF16",
```

```
    "-DENABLE_FP8",
    fp8_macro,
    f"-DSGL_TOPK_DYNAMIC_SMEM_BYTES={topk_dynamic_smem_bytes}",
]

ext_modules = [
    CUDAExtension(
        name="sgl_kernel.common_ops",
        sources=sources,
        include_dirs=include_dirs,
        extra_compile_args={"nvcc": hipcc_flags, "cxx": cxx_flags},
        libraries=libraries,
        extra_link_args=extra_link_args,
        # common_ops 依赖 CPython 专有 API，不能作为 ABI3 扩展发布
        py_limited_api=False,
    ),
]

# 注意：本 PR 移除了 bdist_wheel 的 py_limited_api=cp39 选项，
# 让 wheel 工具生成 CPython 版本专用标签，避免错误声明 ABI3。
setup(
    name="sglang-kernel",
    version=_get_version(),
    packages=find_packages(where="python"),
    package_dir={"": "python"},
    ext_modules=ext_modules,
    cmdclass={"build_ext": BuildExtension.with_options(use_ninja=True)},
)
```

python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce_base.h

QuickReduce 使用的 raw-buffer 指令在 gfx12 上不存在，这里通过空桩让共享扩展在 gfx1250 上仍能编译，同时维持运行时禁用。

```
// llvm.amdgcn.raw.buffer.* 指令在 RDNA4 (gfx12) 上不存在。
// QuickReduce 在 gfx1250 上仍保持运行时禁用；下面的空桩只为了让
// 共享的 ROCm 扩展能编译通过该目标。
#if !defined(__gfx1250__)
// 正常路径：gfx942 / gfx950 等 CDNA 架构上的 raw-buffer 访存内部函数
__quickreduce_device_inline__ static int32x4_t buffer_load_dwordx4(
    int32x4_t srsrc, int32_t voffset, int32_t soffset, int32_t aux) __asm("llvm.amdgcn.raw.buffer.
    load.v4i32");

__quickreduce_device_inline__ static void
buffer_store_dwordx4(int32x4_t data, int32x4_t srsrc, int32_t voffset, int32_t soffset, int32_t
aux) __asm(
    "llvm.amdgcn.raw.buffer.store.v4i32");
#else
// gfx1250 空桩：函数永远不应被调用，因为 QuickReduce 在该架构运行时被禁用。
// 提供空实现只是为了编译通过，避免整个 ROCm 扩展因缺失指令而构建失败。
```

```
__quickreduce_device_inline__ static int32x4_t
buffer_load_dwordx4(int32x4_t srsrc, int32_t voffset, int32_t soffset, int32_t aux) {}

__quickreduce_device_inline__ static void
buffer_store_dwordx4(int32x4_t data, int32x4_t srsrc, int32_t voffset, int32_t soffset, int32_t
aux) {}
#endif
```

评论区精华

无实质技术评审讨论。核心交互发生在合并阶段：maintainer merrymercy 在 CI 长时间未触发后留言 "very weird. The action is not triggered after a long wait. Will merge anyway and fix forward."，作者 oulgen 回复指出 GitHub Actions 当时故障（引用 githubstatus.com）。该 PR 实际在 CI 未通过 / 未运行的情况下直接合并。另有 hubertlu-tw 附带 @akao-amd 提请 AMD 团队关注，无后续技术结论。

- CI 未触发与合并决策 (other): 该 PR 在 CI 未跑的情况下被合并，风险由后续修复承担。
- AMD 团队关注 (question): 无技术结论，没有进一步回复。

风险与影响

- 风险：主要风险集中在打包与编译期行为： 1) gfx1250 下 buffer_load_dwordx4 / buffer_store_dwordx4 变成空操作，若未来有人误在 gfx1250 上启用 QuickReduce，会触发静默错误数据，目前依赖运行时架构门控兜底； 2) wheel 标签从 ABI3 变为 CPython 版本专用，用户在跨 Python 版本复用 wheel 时需要重新安装对应版本； 3) 本次无测试文件配套且 CI 未运行即合并，回归保护依赖事后修复。该变更不触及推理主路径和算子实现。
  - 影响：对 AMD 用户在 gfx1250（RDNA4）环境安装、使用 sgl-kernel 的阻塞被解除；对现有 gfx942/gfx950 用户无运行期影响，仅打包标签更准确。团队侧：这是 AMD 硬件构建 / 发布基础设施的扩展，后续发布流程需要按新 wheel 标签分发，也意味着 ROCm 发布矩阵增加了 gfx1250 目标。
  - 风险标记：构建配置变更，打包标签变更，无新增测试，QuickReduce 空桩需运行时门控兜底，CI 未运行即合并

关联脉络

- PR #33809 [AMD] Move test_load_weights_from_remote_instance.py to extra CI: 同一 AMD 构建 /CI 基础设施方向，都涉及 ROCm 平台的构建与测试组织调整。
- PR #29677 [AMD] perf: compact Triton extend-attention for ragged prefill (AMD/HIP-only): AMD 平台性能与内核调度相关工作，说明仓库对 AMD 支持的持续投入，与 gfx1250 支持形成互补。