PR #50219 完整报告

vllm-project/vllm

[CPU][s390x] Optimize inference perf and add oneDNN INT8 GEMM for s390x

合并时间：2026-07-31 17:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50219

执行摘要

本 PR 为 IBM Z（s390x）CPU 后端补齐向量化推理能力：在 csrc/cpu/cpu_types_vxe.hpp 中新增 FP32Vec16 的 exp/abs/min/clamp/reduce_min 等算子，将 reduce_sum/reduce_max 从标量循环改为基于 vec_add/vec_max/vec_sld 的树形归约，并在构建层打通 oneDNN，使 s390x 首次支持 compressed-tensor INT8 W8A8 GEMM。改动同时涉及 CMake 构建补丁与文档声明，属于 s390x 平台能力的系统性提升，对 x86/ARM 等其他后端无影响；主要风险集中在无测试配套与构建期第三方库补丁的可持续性上。

功能与动机

PR body 明确了目标：为 s390x 应用 VXE 指令集特性——通过 minimax 多项式实现 attention softmax 的快速 exp、用向量指令实现 reduce_sum/reduce_max、用 vec_cts/vec_packs 实现 FP32→INT8 打包，并启用 oneDNN INT8 W8A8 GEMM。此前 vLLM 的 CPU 量化矩阵中 compressed-tensor INT8 W8A8 仅标注 x86 only，本 PR 将其扩展到 s390x，作者以 RedHatAI/granite-3.1-2b-instruct-quantized.w8a8 在 s390x 主机上跑通了端到端推理作为验证。

实现拆解

1. SIMD 原语层补齐（csrc/cpu/cpu_types_vxe.hpp）

- 为 FP32Vec16 新增 exp（委托 FP32Vec8 的 5 阶 minimax 多项式近似）、abs、min、clamp、reduce_min，供 attention softmax 与 dnnl_kernels 使用。
- 新增 INT8Vec16 类型，用 vec_cts/vec_packs 完成 FP32→INT8 打包，服务于 W8A8 GEMM 的量化激活准备。
- BF16Vec16 增加带 elem_num 参数的 save 重载，按实际元素数写出，处理尾部不足 16 个元素的场景。

2. 归约逻辑向量化（csrc/cpu/cpu_types_vxe.hpp）

- FP32Vec8/FP32Vec16 的 reduce_sum 与 reduce_max 从 AliasReg + unroll_loop 的逐元素标量累加，改为 vec_add/vec_max 先合并 128 位块、再用 vec_sld 按 8/4 字节步长折叠的树形归约，最后仅做一次 vec_extract。
- 该改动是性能提升的主要来源之一，同时消除了旧实现中 union 类型重解释带来的潜在 aliasing 问题。

3. 平台宏接线（csrc/cpu/cpu_arch_macros.h）

- 新增 __s390x__ 分支的 DEFINE_FAST_EXP 宏，将 fast_exp/fast_exp_f16 绑定到 FP32Vec16::exp()，与 PowerPC 分支同构，使 attention kernel 的 softmax 自动走向量化 exp。

4. oneDNN 构建与算子注册（cmake/cpu_extension.cmake、csrc/cpu/torch_bindings.cpp）

- cmake 将 S390_FOUND 纳入 oneDNN 构建条件；由于 oneDNN 的 src/cpu/s390x/helpers.h 中 vec_type_t<T> &ALWAYS_INLINE operator+= 在 GCC 14/C++20 下无法编译，采用 FetchContent_Populate + 字符串替换 + add_subdirectory 的方式打构建期补丁。
- torch_bindings.cpp 将 __s390x__ 加入 oneDNN 量化算子注册的预编译条件，使 create_onednn_scaled_mm_handler、onednn_scaled_mm 等 op 在 s390x 下可见。

5. 文档配套

- docs/getting_started/installation/cpu.md：INT8 W8A8 支持范围由 x86 only 更新为 x86、s390x only。
- docs/getting_started/installation/cpu.s390x.inc.md：硬件基线从 Z14 提升到 Z15。

6. 测试配套说明

- 本 PR 未附带直接对应的测试文件改动，验证主要依赖作者在 s390x 主机上的端到端推理日志；CI 侧由维护者触发 Buildkite。

csrc/cpu/cpu_types_vxe.hpp

本 PR 的核心：新增 FP32Vec16 的 exp/abs/min/clamp/reduce_min 算子、INT8Vec16 打包原语，并把 reduce_sum/reduce_max 改为 vec_sld 树形归约，是 s390x 性能提升的主体

```
// FP32Vec8：8 个 FP32 元素的 256 位寄存器组，本 PR 中归约从
// 逐元素标量累加改为向量树形归约，减少指令依赖链长度
struct FP32Vec8 : public Vec<FP32Vec8> {
  f32x4x2_t reg;

  // 先横向相加两个 128 位块，再用 vec_sld 按 8/4 字节步长折叠，
  // 最终只做 1 次标量提取，替代原先的 unroll_loop 逐元素累加
  float reduce_sum() const {
    __vector float sum = vec_add(reg.val[0], reg.val[1]);
    __vector float hi = vec_sld(sum, sum, 8);
    sum = vec_add(sum, hi);
    __vector float lo = vec_sld(sum, sum, 4);
    sum = vec_add(sum, lo);
    return vec_extract(sum, 0);
  }
};

// FP32Vec16：16 个 FP32 元素，由 4 个 128 位块组成
struct FP32Vec16 : public Vec<FP32Vec16> {
  f32x4x4_t reg;
```

```
  // 4 块先两两相加，再按 8/4 字节步长折叠归约，
  // 相比标量循环减少了 15 次加法串行依赖
  float reduce_sum() const {
    __vector float sum = vec_add(vec_add(reg.val[0], reg.val[1]),
                                 vec_add(reg.val[2], reg.val[3]));
    __vector float hi = vec_sld(sum, sum, 8);
    sum = vec_add(sum, hi);
    __vector float lo = vec_sld(sum, sum, 4);
    sum = vec_add(sum, lo);
    return vec_extract(sum, 0);
  }

  // 与 reduce_sum 同构的树形最大归约，用于量化 scale 计算等场景
  float reduce_max() const {
    __vector float m = vec_max(vec_max(reg.val[0], reg.val[1]),
                               vec_max(reg.val[2], reg.val[3]));
    __vector float hi = vec_sld(m, m, 8);
    m = vec_max(m, hi);
    __vector float lo = vec_sld(m, m, 4);
    m = vec_max(m, lo);
    return vec_extract(m, 0);
  }

  // 5 阶 minimax 多项式近似 exp：拆成两个 FP32Vec8 复用其实现，
  // 由 DEFINE_FAST_EXP 宏接入 attention softmax 热路径
  FP32Vec16 exp() const {
    FP32Vec8 lo(f32x4x2_t{reg.val[0], reg.val[1]});
    FP32Vec8 hi(f32x4x2_t{reg.val[2], reg.val[3]});
    auto lo_e = lo.exp();
    auto hi_e = hi.exp();
    return FP32Vec16(f32x4x4_t{lo_e.reg.val[0], lo_e.reg.val[1],
                               hi_e.reg.val[0], hi_e.reg.val[1]});
  }
};
```

csrc/cpu/cpu_arch_macros.h

为 s390x 增加 DEFINE_FAST_EXP 宏分支，使 attention softmax 自动使用向量化近似 exp；也是 review 中架构性质疑的焦点位置

```
// IBM Z（s390x）VXE 扩展：把 attention softmax 的快速 exp 绑定到向量化实现
#ifdef __s390x__
  // FP32Vec16::exp() 在 cpu_types_vxe.hpp 中委托给 FP32Vec8::exp()，
  // 后者用 VXE 内建指令实现 5 阶 minimax 多项式近似，
  // fast_exp_f16 复用同一实现以满足 FP16 路径的调用约定
  #define DEFINE_FAST_EXP \
    auto fast_exp = [&](const vec_op::FP32Vec16& vec) \
                    __attribute__((always_inline)) { return vec.exp(); }; \
    auto fast_exp_f16 = fast_exp;
```

```
#endif // __s390x__
```

评论区精华

> fadara01（cpu_arch_macros.h:190）：“given that your fast exp implementation isn't different form the exp implementation in the VXE class - why don't you just modify cpu_attn_impl.hpp to use vec.exp() for all backends? This would allow us to leverage SIMD exp implementations and speed up other backends too.”——这是本 PR 最有价值的讨论：DEFINE_FAST_EXP 宏本质是跨后端的重复样板，收敛为统一调用 vec.exp() 可以让 x86/ARM 等后端也受益，PR#50133 正是在做这件事。作者未回复，但 reviewer 在批准合并的同时保留了该演进方向。
>
> depthfirst-app[bot]（LOW）：INT8Vec16::save 的循环缺少 i < VEC_ELEM_NUM 上界检查，与本文件其他 save(ptr, elem_num) 的防御模式不一致，elem_num > 16 时会造成栈上越界读与目标缓冲区越界写。建议补上界检查。

风险与影响

- 越界风险（INT8Vec16::save）：bot 指出的越界问题没有作者回复证据，合并后应抽查确认该函数是否带 i < VEC_ELEM_NUM 守卫；若调用方总能保证 elem_num ≤ 16 则无害。
- oneDNN 源码补丁脆弱：cmake 构建期改写 oneDNN 的 helpers.h 属于硬编码字符串替换，oneDNN 版本升级后可能静默失效或产生难懂的错误；后续应推动上游修复或改用 ONEDNN_ 编译选项替代。
- 无自动化测试覆盖：INT8 W8A8 在 s390x 上的正确性仅靠手工端到端日志验证，回归风险由后续 CI 守护承担。
- 近似精度差异：FP32Vec16::exp 为最小最大多项式近似，softmax 输出与 libm 版本存在 epsilon 级偏差，对严格数值对齐场景（如对照实验）需留意。
- 硬件基线收紧：文档将 s390x 要求从 Z14 提升为 Z15，Z14 用户可能在运行时遭遇指令不支持。

关联脉络

- PR#50133（review 中引用）为“缺失的向量化后端补齐 exp”，与本 PR 的 DEFINE_FAST_EXP 属于同一功能演进线；两者并读可以看到 vLLM CPU backend 正在把 SIMD 近似数学函数从各架构各自实现，逐步收敛为统一策略。
- 从历史 PR 看，CPU 平台的能力补齐通常分两步走：先做 SIMD 原语（本 PR），再补构建/CI 矩阵与测试（如 XPU、ROCm 系列的 CI 调整 PR）。本 PR 缺少测试配套，后续大概率会有针对 s390x 量化路径的 CI 或单元测试 PR 跟进。
- 构建期第三方库补丁的做法（FetchContent_Populate + 字符串替换）与 #50328（CMake 引入 triton_kernels）同属 CPU/ROCm 构建链路的定制化演进，说明 vLLM 对异构 CPU 架构的构建适配仍处于快速迭代期。