PR #47105 完整报告

vllm-project/vllm

[XPU] Support ZE_AFFINITY_MASK passthrough in xpu_disagg_acc_test

合并时间：2026-07-01 01:06

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47105

执行摘要

- 一句话：XPU 解耦测试支持 ZE_AFFINITY_MASK 透传
- 推荐动作：该 PR 为低风险的基础设施优化，建议合并。新增独立 CI job 的设计值得在其他场景推广。

功能与动机

允许 XPU disagg accuracy 测试通过外部传递 ZE_AFFINITY_MASK 进行设备选择，以适配不同 GPU 配置环境。

实现拆解

1. 重构设备亲和性逻辑：在 run_xpu_disagg_accuracy_test.sh 中，将原有的 generate_affinity_mask 函数替换为基于 ZE_AFFINITY_MASK 环境变量的解析与索引机制。如果设置了 ZE_AFFINITY_MASK，则将其解析为可用设备列表 AVAILABLE_DEVICES，否则按默认顺序生成设备 ID。新增 compute_gpu_id 函数，根据起始索引和 TP 大小从 AVAILABLE_DEVICES 中提取对应的 GPU ID 字符串。
2. 更新启动函数：在 launch_baseline 和 launch_pd 函数中，使用 compute_gpu_id 动态计算 ZE_AFFINITY_MASK 的值，替换原来的硬编码 0 或独立变量 PREFILLER_ZE_AFFINITY_MASK / DECODER_ZE_AFFINITY_MASK。同时将环境变量名 VLLM_MULTIPROC_EXECUTE_MODEL_TIMEOUT_S 改为 VLLM_EXECUTE_MODEL_TIMEOUT_SECONDS。
3. 拆分 CI job：在 .buildkite/intel_jobs/misc_intel.yaml 中新增一个独立的 job NixlConnector PD accuracy (2 GPUs)，专门运行该测试脚本，并配置了 2 个 GPU 资源、num_devices: 2。同时从 .buildkite/intel_jobs/test-intel.yaml 的 "XPU V1 test" job 中移除该测试的调用行，实现关注点分离。

关键文件：

- tests/v1/kv_connector/nixl_integration/run_xpu_disagg_accuracy_test.sh（模块 测试脚本；类别 test；类型 test-coverage）：核心变更文件，重写设备亲和性逻辑以支持 ZE_AFFINITY_MASK 透传。
- .buildkite/intel_jobs/misc_intel.yaml（模块 CI 配置；类别 config；类型 configuration）：新增独立的 CI job 用于 XPU disagg accuracy 测试，分离关注点。

- .buildkite/intel_jobs/test-intel.yaml（模块 CI 配置；类别 config；类型 configuration）：从原有 XPU V1 test 中移除独立的测试调用，配合拆分。

关键符号：未识别

关键源码片段

tests/v1/kv_connector/nixl_integration/run_xpu_disagg_accuracy_test.sh

核心变更文件，重写设备亲和性逻辑以支持 ZE_AFFINITY_MASK 透传。

```
# 解析 ZE_AFFINITY_MASK（例如 "2,3"）或默认使用 "0,1,..."。
# compute_gpu_id 根据索引从此列表中为每个实例分配设备。
if [[ -n "${ZE_AFFINITY_MASK:-}" ]]; then
  # 以逗号分割用户提供的掩码
  IFS=',' read -r -a AVAILABLE_DEVICES <<< "${ZE_AFFINITY_MASK}"
else
  # 默认生成 0 到 (PREFILLER_TP_SIZE + DECODER_TP_SIZE - 1) 的设备列表
  AVAILABLE_DEVICES=()
  for ((i=0; i<PREFILLER_TP_SIZE + DECODER_TP_SIZE; i++)); do
    AVAILABLE_DEVICES+=("$i")
  done
fi

# 从 AVAILABLE_DEVICES 中提取连续的 GPU ID 字符串
compute_gpu_id() {
  local start=$1
  local tp_size=$2
  local gpu_id="${AVAILABLE_DEVICES[$start]}"
  for (( j=1; j<tp_size; j++ )); do
    gpu_id="${gpu_id},${AVAILABLE_DEVICES[$((start + j))]}"
  done
  echo "${gpu_id}"
}

# 在 launch_baseline 和 launch_pd 中使用 compute_gpu_id 动态计算 ZE_AFFINITY_MASK
launch_baseline() {
  local BASELINE_GPU_ID
  BASELINE_GPU_ID=$(compute_gpu_id 0 1) # baseline 始终使用第一个设备
  BASELINE_BASE_CMD="
  ZE_AFFINITY_MASK=$BASELINE_GPU_ID \
  VLLM_WORKER_MULTIPROC_METHOD=spawn \
  VLLM_ENABLE_V1_MULTIPROCESSING=1 vllm serve $MODEL_NAME \
      --host ${BASELINE_HOST} \
      --port ${BASELINE_PORT} \
      --max-model-len ${MAX_MODEL_LEN}\
      --seed 42 \
      -tp 1 \
      --block-size ${BLOCK_SIZE} \
      --gpu-memory-utilization ${GPU_MEMORY_UTILIZATION} \
      --dtype float16 \
```

```
      --enforce-eager"
  echo ${BASELINE_BASE_CMD}
  bash -c "${BASELINE_BASE_CMD}" &
  sleep 10
  wait_for_server ${BASELINE_HOST} ${BASELINE_PORT}
}

launch_pd() {
  local PREFILL_GPU_ID
  PREFILL_GPU_ID=$(compute_gpu_id 0 "${PREFILLER_TP_SIZE}")
  local DECODE_GPU_ID
  DECODE_GPU_ID=$(compute_gpu_id "${PREFILLER_TP_SIZE}" "${DECODER_TP_SIZE}")
  # ... 在命令中使用 $PREFILL_GPU_ID 和 $DECODE_GPU_ID
}
```

评论区精华

该 PR 无 review 评论讨论。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更仅限于测试脚本和 CI 配置，不涉及核心代码。但需要确保新 job 的依赖文件和资源标签正确，避免 CI 调度失败。
- 影响：影响范围限于 XPU CI 测试。测试的隔离使得 disagg accuracy 测试不再绑定于 V1 常规测试，便于独立调试和资源分配。
- 风险标记：CI 配置变更，仅测试脚本改动

关联脉络

- PR #47157 [CI][Bugfix] Fix Hybrid SSM NixlConnector PD prefix cache test (2 GPUs)：同属 NixlConnector PD 测试的 CI 修复，均在 XPU/intel GPU 环境下运行。