PR #21247 完整报告

sgl-project/sglang

[Dependency] Upgrade to Torch 2.11.0

合并时间：2026-05-03 03:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21247

执行摘要

- 一句话：升级 PyTorch 至 2.11.0 并适配 CI 与模型兼容性
- 推荐动作：建议所有开发者更新本地环境至 PyTorch 2.11 以与主分支对齐；重点审查扩散模型的输出差异和 MoE 编译 guard 逻辑。

功能与动机

根据 PyTorch 2.11.0 发布说明，升级以获得新特性、性能改进和 bug 修复，并保持与上游同步。PR body 直接引用发布说明链接。

实现拆解

1. 更新 pyproject.toml 中的 torch 版本约束，并升级 sgl-kernel 到 0.4.2 以匹配新 ABI（commit 6bcfd91bdc）。
2. 在 CI 工作流中强制重建 deep_ep（FORCE_REBUILD_DEEPEP），避免旧缓存二进制与 torch 2.11 ABI 冲突；同时清理 cu12 后缀包的冲突设置。
3. 移除过时的 NVIDIA wheel 缓存脚本 scripts/ci/cuda/cache_nvidia_wheels.sh，简化依赖安装流程；调整 jit-kernel 和 NPU 测试工作流以复用 sgl-kernel 构建产物。
4. 修复 transformers 库的 RobertaProcessing API 变更：在 component_loader.py 中添加 _load_auto_tokenizer_with_roberta_processing_compat，在 test_utils.py 中添加对应的 CLIP 处理器回退，确保扩散模型加载不崩溃。
5. 调低 diffusion consistency 测试阈值（consistency_threshold.json），并更新测试代码以在失败时保存可视化 artifact，适应 torch 2.11 下的轻微数值变化。

关键文件：

- python/sglang/multimodal_gen/runtime/loader/component_loaders/component_loader.py（模块 加载器；类别 source；类型 core-logic；符号 _load_auto_tokenizer_with_roberta_processing_compat, roberta_processing_compat）：核心逻辑：新增 _load_auto_tokenizer_with_roberta_processing_compat 函数，以兼容 tokenizers>=0.21 移除 cls 参数的变更，确保扩散模型中的 CLIP tokenizer 加载不崩溃。
- python/sglang/multimodal_gen/test/test_utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 _load_clip_processor_with_roberta_processing_compat, roberta_processing_compat, _safe_artifact_name, _format_metric_value）：测试配套：新增 CLIP 处理器兼容函数 _load_clip_processor_with_roberta_processing_compat、

failure artifact 保存相关辅助函数，以及修改 get_clip_model 以捕获 RobertaProcessing 错误。

- python/sglang/srt/models/deepseek_v2.py（模块 深搜模型；类别 source；类型 data-contract）：MoE forward guard 调整：扩展 torch.compile 跳过条件，考虑 speculative_num_draft_tokens，避免 EAGLE/MTP 场景下错误使用 dual_stream 路径。
- .github/workflows/pr-test-npu.yml（模块 CI 配置；类别 infra；类型 infrastructure）：CI 配置：大幅修改 NPU 测试工作流，增加缓存、依赖安装步骤，适配 torch 2.11 环境。

关键符号：_load_auto_tokenizer_with_roberta_processing_compat, roberta_processing_compat, _load_clip_processor_with_roberta_processing_compat, _safe_artifact_name, _format_metric_value, _json_metric_value, _metric_items, _text_width, _resize_for_comparison, test_save_consistency_failure_artifact, load_customized (modified), get_clip_model (modified), forward (deepseek_v2, modified)

关键源码片段

python/sglang/multimodal_gen/runtime/loader/component_loaders/component_loader.py

核心逻辑：新增 _load_auto_tokenizer_with_roberta_processing_compat 函数，以兼容 tokenizers>=0.21 移除 cls 参数的变更，确保扩散模型中的 CLIP tokenizer 加载不崩溃。

```
def _load_auto_tokenizer_with_roberta_processing_compat(*args, **kwargs):
    # tokenizers>=0.21 移除了 RobertaProcessing 的 `cls` 参数，
    # 但 transformers 的 CLIPTokenizer 构建仍会传递该参数。
    # 此函数在加载前临时替换 processors.RobertaProcessing 为兼容包装，
    # 并确保加载后恢复原值，避免影响其他调用。
    from tokenizers import processors

    roberta_processing = processors.RobertaProcessing

    def roberta_processing_compat(*processor_args, **processor_kwargs):
        if "sep" in processor_kwargs and "cls" in processor_kwargs:
            sep = processor_kwargs.pop("sep")
            cls_token = processor_kwargs.pop("cls")
            return roberta_processing(
                sep, cls_token, *processor_args, **processor_kwargs
            )
        return roberta_processing(*processor_args, **processor_kwargs)

    processors.RobertaProcessing = roberta_processing_compat
    try:
        return AutoTokenizer.from_pretrained(*args, **kwargs)
    finally:
        processors.RobertaProcessing = roberta_processing


class TokenizerLoader(ComponentLoader):
```

```
    ...
    def load_customized(self, component_model_path, server_args, component_name):
        ...
        try:
            return AutoTokenizer.from_pretrained(...)
        except TypeError as e:
            if "RobertaProcessing" in str(e) and use_fast:
                logger.warning("Fast tokenizer failed, retrying with compat...")
                # 使用兼容包装重新加载
                return _load_auto_tokenizer_with_roberta_processing_compat(
                    component_model_path,
                    padding_side="right",
                    use_fast=False,
                )
            raise
```

python/sglang/srt/models/deepseek_v2.py

MoE forward guard 调整：扩展 torch.compile 跳过条件，考虑 speculative_num_draft_tokens，避免 EAGLE/MTP 场景下错误使用 dual_stream 路径。

```
# 在 MoE 的 forward 中，当启用 torch.compile 并且当前 batch 大小
# 小于等于 torch_compile_max_bs * draft_tokens 时，跳过 dual_stream 路径，
# 避免 torch.compile 生成的计算图与 EAGLE/MTP 共享导致的数值错误。
if (
    self.num_fused_shared_experts == 0
    and hidden_states.shape[0] > 0
    and get_is_capture_mode()
    and not (
        get_global_server_args().enable_torch_compile
        and hidden_states.shape[0]
        <= get_global_server_args().torch_compile_max_bs
        * (get_global_server_args().speculative_num_draft_tokens or 1)
    )
):
    return self.forward_normal_dual_stream(
        hidden_states,
        ...
    )
```

评论区精华

无实质技术讨论；评论主要涉及 CI 重新触发命令。PR 的合并者通过多次 rerun 确保各平台测试通过。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险在于 torch 2.11 可能引入 API 变更导致运行时崩溃，已通过强制重建 deep_ep、添加 RobertaProcessing 回退等方式缓解；扩散模型数值精度变化已通过调整阈值适应；但仍有潜在的性能回归未被完全验证，尤其是 torch.compile 相关路径。
- 影响：影响所有基于 PyTorch 2.11 运行的 SGLang 服务；CI 流程更简洁，但要求 runner 具备新版本环境；AMD、NPU、Blackwell 等平台的测试均需通过；扩散模型一致性基准需要重新校准。
- 风险标记：依赖版本冲突，ABI 兼容性，扩散数值偏移，MoE guard 正确性

关联脉络

- 暂无明显关联 PR