PR #6992 完整报告

PaddlePaddle/FastDeploy

[Feature] Added the /v1/abort_requests endpoint

合并时间：2026-03-31 11:02

原文链接：http://prhub.com.cn/PaddlePaddle/FastDeploy/pull/6992

执行摘要

- 一句话：新增主动中断推理请求的 /v1/abort_requests 端点，支持中止特定或全部请求。
- 推荐动作：建议精读此 PR 以理解主动控制接口的设计模式，特别关注并发安全和 API 一致性讨论。对于类似功能开发，可参考其实现，但需注意修复 review 中提出的风险点，如加锁保护、统一参数名和补充测试。

功能与动机

PR body 中说明：'Currently, the logic for interrupting requests and stopping inference can only be triggered by the client disconnecting. Since there is no interface for active triggering, this new endpoint is required to encapsulate and expose the existing internal capabilities.' 即需提供主动触发接口来封装和暴露现有内部能力。

实现拆解

1. API 层：在 fastdeploy/entrypoints/openai/api_server.py 和 fastdeploy/router/router.py 新增 /v1/abort_requests 端点，处理 HTTP 请求并转发控制方法。
2. Engine 层：在 fastdeploy/engine/common_engine.py 新增 _control_abort_requests 方法，根据参数 abort_all 或 req_ids 确定目标请求列表，收集部分结果并触发中止逻辑，包括调用 resource_manager.add_abort_req_ids。
3. 资源管理：在 fastdeploy/engine/sched/resource_manager_v1.py 新增 get_reqs_in_aborting 方法，支持查询中止中请求状态。
4. 响应处理：在 fastdeploy/entrypoints/openai/serving_chat.py 和 serving_completion.py 中，根据错误消息标记 aborted 请求的 finish_reason 为 'abort'。
5. 文档更新：在 docs/online_serving/README.md 和 router.md 中添加端点说明和使用示例。
6. 测试：新增测试文件覆盖 api_server、router 和 engine 的功能。

关键文件：

- fastdeploy/entrypoints/openai/api_server.py（模块 APIServer）：新增 /v1/abort_requests 端点，处理 API 请求并调用 engine 控制方法，是用户直接交互的接口。

- fastdeploy/router/router.py（模块 Router）：新增 /v1/abort_requests 端点，转发请求到所有服务器并聚合结果，支持分布式环境下的请求中止。
- fastdeploy/engine/common_engine.py（模块 Engine）：实现 _control_abort_requests 方法，包含核心中止逻辑、并发处理和部分结果收集，是功能的核心实现。
- fastdeploy/engine/sched/resource_manager_v1.py（模块 Scheduler）：新增 get_reqs_in_aborting 方法，支持查询中止中请求状态，是资源管理的关键扩展。
- tests/engine/test_common_engine.py（模块 Testing）：新增测试覆盖 _control_abort_requests 方法，验证功能正确性和边界条件，但 review 指出测试可能不足。

关键符号：_control_abort_requests, abort_requests, get_reqs_in_aborting

评论区精华

1. 参数命名争议：mitu626 评论 'abort_all 还是 stop_all'，Copilot 指出文档使用 stop_all 而实现使用 abort_all，导致 API 不一致，建议统一字段名。
2. 并发安全风险：Copilot 评论在 _control_abort_requests 中遍历 resource_manager.requests 和 scheduler.requests 时未加锁，可能抛出 'dictionary changed size during iteration' 异常；get_reqs_in_aborting 方法中集合运算也存在并发问题，建议加锁或使用线程安全接口。
3. 测试覆盖不足：Copilot 建议为 router 和 api_server 的新端点补充单元测试，以确保关键行为（如转发逻辑、参数校验）得到覆盖。
4. 协议兼容性问题：Copilot 指出 finish_reason 值 'abort' 不在协议定义中，可能导致 Pydantic 校验错误，建议修改为合法值或更新协议。
5. 超时设置缺失：Copilot 评论 router 端请求转发时未设置超时，可能导致线程挂起，建议添加超时配置。

- 参数命名一致性 (design): 未解决，建议统一字段名（如使用 abort_all）并更新文档。
- 并发安全风险 (correctness): 建议在 resource_manager 和 scheduler 中加锁保护或使用线程安全接口，但当前实现未修复。
- 测试覆盖不足 (testing): PR 中已添加部分测试（如 test_common_engine.py），但 router 端测试可能不充分，建议进一步补充。
- 协议兼容性问题 (correctness): 未解决，需调整代码或协议以避免运行时错误。

风险与影响

- 风险：1. 并发安全风险：在 fastdeploy/engine/common_engine.py 的 _control_abort_requests 方法中，直接遍历 self.resource_manager.requests.keys() 和 self.scheduler.requests.keys()，未加锁保护，在多线程环境下可能引发运行时错误。 2. API 不一致风险：文档中参数名为 stop_all，而代码实现使用 abort_all，客户端调用时可能导致 400 错误或功能失效，影响用户体验。 3. 协议错误风险：finish_reason 值 'abort' 未在协议文件（如 protocol.py）中定义，非流式路径中可能触发 Pydantic 校验失败，导致接口返回 500 错误。 4. 测试覆盖不足风险：尽管已添加部分测试，但 router 端和 api_server 端的新端点测试可能不充分，回归风险较高。 5. 超时风险：router 端请求转发时未设置超时，当服务器不可达时可能导致请求挂起，影响系统稳定性。

- 影响：1. 用户影响：提供了主动中断推理请求的能力，便于资源管理和调试，但需注意 API 参数名可能变更，需更新客户端代码。 2. 系统影响：增强了控制平面功能，可释放 GPU 内存（KV Cache）和计算资源，可能影响现有请求调度和资源回收逻辑。 3. 团队影响：工程师需关注并发安全和 API 设计一致性，后续维护中需解决 review 中未决的疑虑，如参数命名和协议更新。
- 风险标记：并发安全风险，API 不一致，缺少测试覆盖，协议错误

关联脉络

- PR #7042 [RL] Adapt async rollout checkpoint update flow: 同样涉及控制接口（update_weights）的修改，共享设计模式和 APIServer 集成，可作为参考。