PR #25101 完整报告

sgl-project/sglang

[radix cache] pluggable RadixCache factory (--radix-cache-backend)

合并时间：2026-05-21 01:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25101

执行摘要

- 一句话：引入可插拔 RadixCache 后端注册机制
- 推荐动作：该 PR 设计简洁、测试充分，值得阅读。尤其是注册机制与默认回退的组合模式，可以复用于其他需要类似扩展点的子系统。测试中的 _RegistryIsolationMixin 也是处理全局状态隔离的良好示例。

功能与动机

PR 描述指出：‘Similar to other backends, add a registry-based extension point for radix-cache, which would be more friendly for non-OSS Radix cache developments.’目标是为非开源 Radix Cache 开发提供友好的扩展点，同时不改变默认行为。

实现拆解

实现分以下几步：

1. 创建 registry.py：定义 TreeCacheBuildContext 数据类封装所有构造参数，RadixCacheFactory 类型别名，以及 register_radix_cache_backend、get_radix_cache_factory 等注册函数。内置选择链被提取为 default_radix_cache_factory，按优先级依次尝试 chunk cache、CPP 实现、统一缓存、分层缓存、SWA/SSM 专用缓存、lmcache 和默认 RadixCache。
2. 简化 kv_cache_builder.py：将原来多达 80 行的内联 if-elif 链替换为对 create_tree_cache(TreeCacheBuildContext(...)) 的单次调用，同时移除不再需要的局部导入（如 StreamingSession、各个 Cache 类）。导入移到文件顶部。
3. 添加 CLI 参数：在 server_args.py 中新增 radix_cache_backend 字段和 --radix-cache-backend 参数，接受注册过的后端名字。
4. 编写单元测试：新增 test/registered/unit/mem_cache/test_registry.py，包含注册验证、重复 / 空名异常、路由分发（注册、未注册、默认回退）、streaming session 自动包装 / 不包装逻辑。测试使用 _RegistryIsolationMixin 在每个用例前后保存 / 恢复全局 registry 快照，保证隔离。

关键文件：

- python/sglang/srt/mem_cache/registry.py（模块 缓存层；类别 source；类型 dependency-wiring；符号 TreeCacheBuildContext, register_radix_cache_backend, get_radix_cache_factory, registered_radix_cache_backends）：新增的注册中心文件，

定义了 TreeCacheBuildContext、register_radix_cache_backend、default_radix_cache_factory、create_tree_cache 等核心函数，是整个 PR 的架构核心。

- test/registered/unit/mem_cache/test_registry.py（模块 测试用例；类别 test；类型 test-coverage；符号 _make_ctx, _RegistryIsolationMixin, TestRegisterRadixCacheBackend, TestCreateTreeCacheRouting）：全面的单元测试，覆盖注册、路由、异常和 streaming 包装逻辑，确保质量。
- python/sglang/srt/mem_cache/kv_cache_builder.py（模块 缓存层；类别 source；类型 dependency-wiring）：修改了 tree_cache 创建逻辑，从内联条件分支改为调用 create_tree_cache，大幅减少代码量，清晰化依赖。
- python/sglang/srt/server_args.py（模块 配置层；类别 source；类型 core-logic）：新增 radix_cache_backend 配置字段和 CLI 参数，为用户提供选择后端的接口。

关键符号：register_radix_cache_backend, get_radix_cache_factory, registered_radix_cache_backends, default_radix_cache_factory, create_tree_cache, TreeCacheBuildContext

评论区精华

Review 中 MerryMercy 提出了两点改进：

- 在 kv_cache_builder.py 中将局部导入移到文件顶部（避免函数内热路径的重复导入）。
- 在 registry.py 的 TreeCacheBuildContext 中，server_args 的注解可以去掉引号，因为已经使用 TYPE_CHECKING。 两个建议均由作者 Jialin 在后续提交中采纳。
- Import placement in kv_cache_builder.py (style): 作者将 import 移到了顶部。
- Type annotation in TreeCacheBuildContext (style): 作者移除了引号。

风险与影响

- 风险：
  - 核心路径变更：build_kv_cache 是每个模型初始化时必经的路径，替换为 create_tree_cache 后，任何 registry 或 factory 中的错误都会导致启动失败。
  - 全局状态隔离：_RADIX_CACHE_REGISTRY 是模块级全局字典，多测试间如果不隔离可能互相干扰。测试已使用 snapshot 恢复机制，但源码启动时如果多个组件调用 register_radix_cache_backend 需注意命名冲突。
  - 未知后端处理：create_tree_cache 在遇到未注册的后端名字时会抛出 ValueError，用户需确保名字匹配。
  - Streaming session 包装：如果注册的 factory 返回的 cache 不支持 streaming，但 enable_streaming_session=True，则会自动包装 StreamingSession，这可能导致意外行为？但内置逻辑也是如此，所以风险一致。
- 影响：
  - 用户影响：无需任何改动，默认行为保持不变。高级用户可通过注册自定义后端扩展现有缓存策略。

- 系统影响：代码可维护性提升，新增 RadixCache 实现不再需要修改 kv_cache_builder.py，只需注册即可。
- 团队影响：降低了为特定场景（非开源、硬件加速等）开发 RadixCache 的门槛。
- 风险标记：核心路径变更，全局注册表隔离，新 CLI 参数误用

关联脉络

- PR #25770 [Bug][Mamba] Fix LRU list reference cycle leak in mamba_radix_cache: 同样涉及 mem_cache 目录的修改，修复后为本 PR 的抽象化提供了更稳定的基础。
- PR #25819 disagg prebuilt: drop dead prepare_for_extend shift: 对 radix cache 相关代码的清理，本 PR 在此基础上统一了创建入口。