# ZYAgent Pipeline 架构设计

ZYAgent Team

2026 年 1 月 23 日

## 1 主进程架构

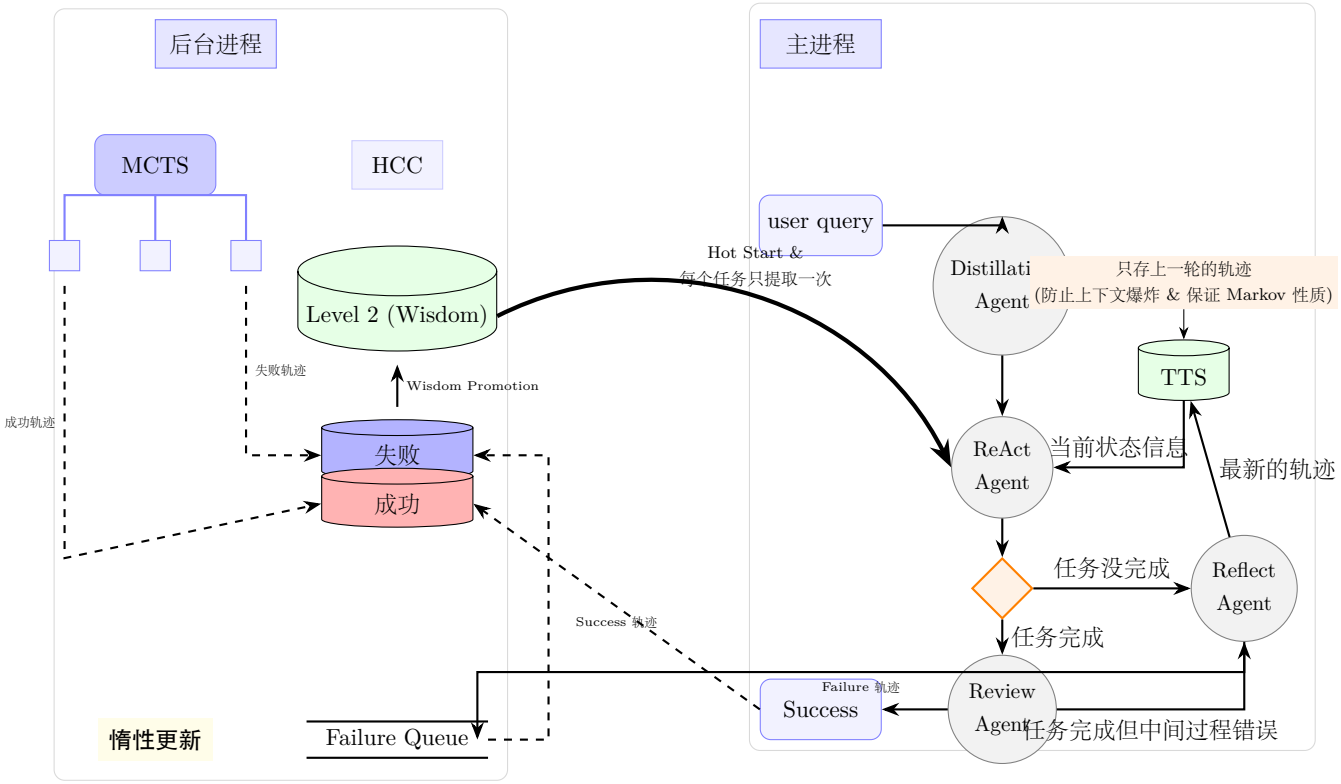


图 1: ZYAgent 最终架构逻辑图：清晰展示后台进程 (MCTS/HCC) 与主进程 (ReAct Loop) 的交互与数据流向

## 2 架构概览 (Overview)

ZYAgent 采用一种创新的 **双系统架构 (Dual-System Architecture)**，旨在模仿人类认知中的快思考 (System 1) 与慢思考 (System 2) 机制。

- **System 1 (主进程)**: 负责实时交互与任务执行，强调响应速度与上下文效率。

- **System 2 (后台进程)**: 负责深度探索与知识沉淀，利用空闲算力进行反事实推理 (Counterfactual Reasoning)。

两者通过 **HCC (Hierarchical Cognitive Cache)** 和异步消息队列进行解耦，确保了系统的高鲁棒性与持续进化能力。

# 3 主进程详解 (Main Process)

主进程是用户直接交互的界面，核心是一个基于 **Markov 性质**的 ReAct 闭环。

1. **Distillation Agent (顶层规划)**:
   - 作为系统的入口，接收 User Query。
   - 负责 **Hot Start**: 从 HCC Level 2 中提取与当前任务最相关的 Wisdom（一次性注入），为 ReAct Agent 提供高质量的初始指导。
2. **ReAct Agent (执行引擎)**:
   - 执行标准的 Thought-Action-Observation 循环。
   - **Context Constraint**: 它的上下文输入严格受限于 TTS Buffer，不依赖完整的历史对话记录。
3. **TTS Buffer (瞬时轨迹存储)**:
   - **设计哲学**: 为了防止 Long-horizon 任务中的上下文爆炸 (Context Explosion)，TTS 仅存储 **上一轮**的关键轨迹信息。
   - **Markov 假设**: 假设当前状态 (State $t$) 包含了决策所需的所有必要信息，从而将推理复杂度从 $O(N^2)$ 降低到 $O(1)$。
4. **Review & Reflect (质量控制闭环)**:
   - **Review Agent**: 检查 ReAct 的输出是否达成目标。如果成功，生成 Success 轨迹；如果失败，触发 Reflect。
   - **Reflect Agent**: 分析失败原因。
     - **短期修正**: 更新 TTS Buffer，将反思结果作为 Context 注入下一轮 ReAct。
     - **长期记忆**: 将难以解决的 Failure 轨迹发送至后台 Failure Queue，寻求 System 2 的帮助。

# 4 后台进程详解 (Background Process)

后台进程作为系统的“潜意识”，在不阻塞主线任务的情况下进行深度学习。

- **Failure Queue (失败队列)**: 接收来自主进程的失败案例。这些是 System 1 无法处理的“难题”。
- **MCTS Simulator (蒙特卡洛树搜索)**:
  - 对 Failure Queue 中的案例进行 **Off-policy** 探索。
  - 通过大量的模拟与回溯，寻找被 ReAct 忽略的成功路径。
- **HCC (分层认知缓存)**:
  - **Level 1 (Samples)**: 存储 MCTS 搜索生成的原始数据（具体的 Success/Failure 轨迹）。
  - **Wisdom Promotion**: 通过算法将 L1 的具体样本提炼为 L2 的通用 **Wisdom**（例如：“在处理网络错误时，应优先检查 DNS 配置而非立即重试”）。
- **Lazy Update (惰性更新)**: 后台的知识更新是异步的。主进程不会等待 MCTS 的结果，而是“惰性”地在下一次任务启动 (Hot Start) 时享受知识更新带来的红利。

# 5 核心机制总结

**Markov Property** 通过限制 Context Window 为 1，强制模型学会信息压缩，极大降低推理成本。

**Hot Start** “每个任务只提取一次”，避免了 RAG 系统中频繁检索带来的噪声与延迟。

**Failure as Fuel** 失败不再是终点，而是进化的养料。每一个 Failure 最终都会转化为 HCC 中的一条 Success Pattern。