AIOS：大模型智能体操作系统

一. 背景介绍

随着大语言模型（LLMs）的爆火，拥有大语言模型充当"大脑"的智能体（agent）出现在各种应用中。借助大语言模型的能力，智能体能够在无需人类干预的前提下，独立运行、做出决策并执行任务，解决复杂问题。

然而，假设我们有一个自己的 LLM，而后有一堆需要 LLM 执行的任务，且我们以 agent 的形式完成这些任务，因为一个智能体通常只执行一个专门的任务，随着我们要执行任务的增多，智能体数量和复杂性就会指数增长。但是我们的 LLM 和操作系统（OS）却只有一个，类似于一个应用，当用户数增加时，其后端访问压力就会越来越大，当 agent 增加时，我们就需要想办法优化"后端"—— LLM 和操作系统。

相比单个 LLM 和单个 OS 上以应用软件的形式运行 agent 可能遇到的问题：

1. 智能体请求在 LLM 上的调度优化和资源分配；（agent 所执行的任务有优先级之分，如何合理调度）
2. 智能体与 LLM 交互过程中维护上下文的困难；（LLM 单次输入的句子长度有限，多个 agent 该如何共用这些上下文长度）
3. 以及将具有不同能力和专长的异构智能体集成带来复杂性问题；（需要使用多个 agent 时，不同 agent 可能有不同的输入输出，有不同的限制，因此将多个 agent 集成在一起的难度很大）。

而大模型智能体操作系统 AIOS 旨在**优化资源分配，促进智能体之间的上下文切换，实现智能体的并发执行，为智能体提供工具服务，并维护智能体的访问控制。**

二. 整体架构

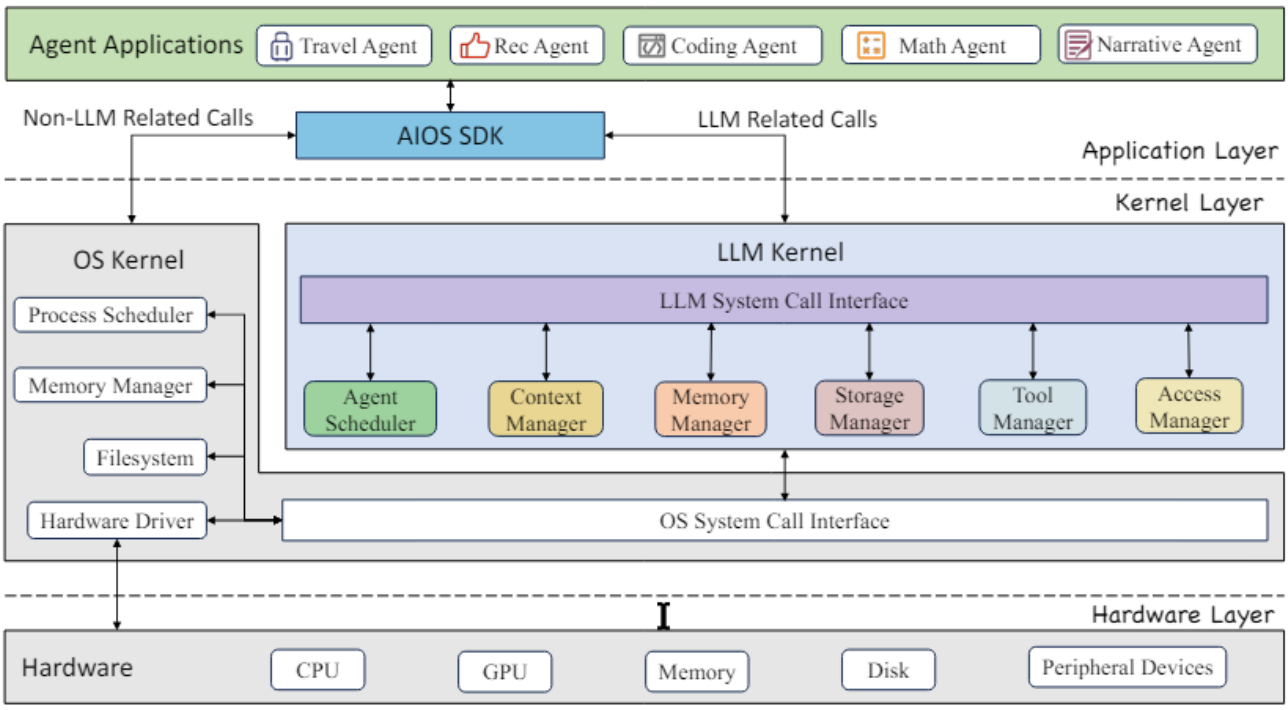

Figure 2: An overview of the AIOS architecture.

如上图所示，AIOS 架构分为三层：**应用层**、**内核层**和**硬件层。**

1. **应用层**： 在应用层，可以开发和部署智能体应用程序。在这一层，AIOS 提供了 AIOS SDK，对系统调用进行了更高级的抽象，从而简化了智能体的开发过程。该 SDK 通过提供丰富的工具包，来抽象低层系统功能的复杂性，使开发者可以专注于智能体的基本逻辑和功能，从而提高开发效率。

2. **内核层**： 内核层分为两个主要部分：操作系统内核和LLM内核，原始的OS内核负责硬件管理，处理无智能软件的请求，而新增加的LLM内核则专用于处理与LLM智能体、LLM相关资源和开发工具包相关的职责。

3. **硬件层**： 硬件层由 CPU、GPU、内存、磁盘和外围设备等组成。硬件资源仍然由操作系统管理。LLM 层仍然调用 OS 提供的底层接口。

在新增加的 LLM 内核中，关键部件如下：

- **智能体调度器**：对智能体请求进行优先级排序和调度，以优化 LLM 资源利用率。
- **上下文管理器**：支持对LLM的中间生成状态进行快照和恢复，以及 LLM 的上下文窗口管理。
- **内存管理器**：为每个智能体的交互日志提供短期内存。
- **存储管理器**：将智能体的交互日志持久化到长期存储中，以便将来检索。
- **工具管理器**：管理智能体对外部 API 工具（如搜索、科学计算）的调用。
- **访问管理器**：在智能体之间强制执行隐私和访问控制策略。

三. 可能的选题方向

1. **智能体调度器**：

我们把每一个智能体（agent）都看成一个进程，这个调度器的作用就显而易见——进程调度器，但是调度的是 agent 的请求。

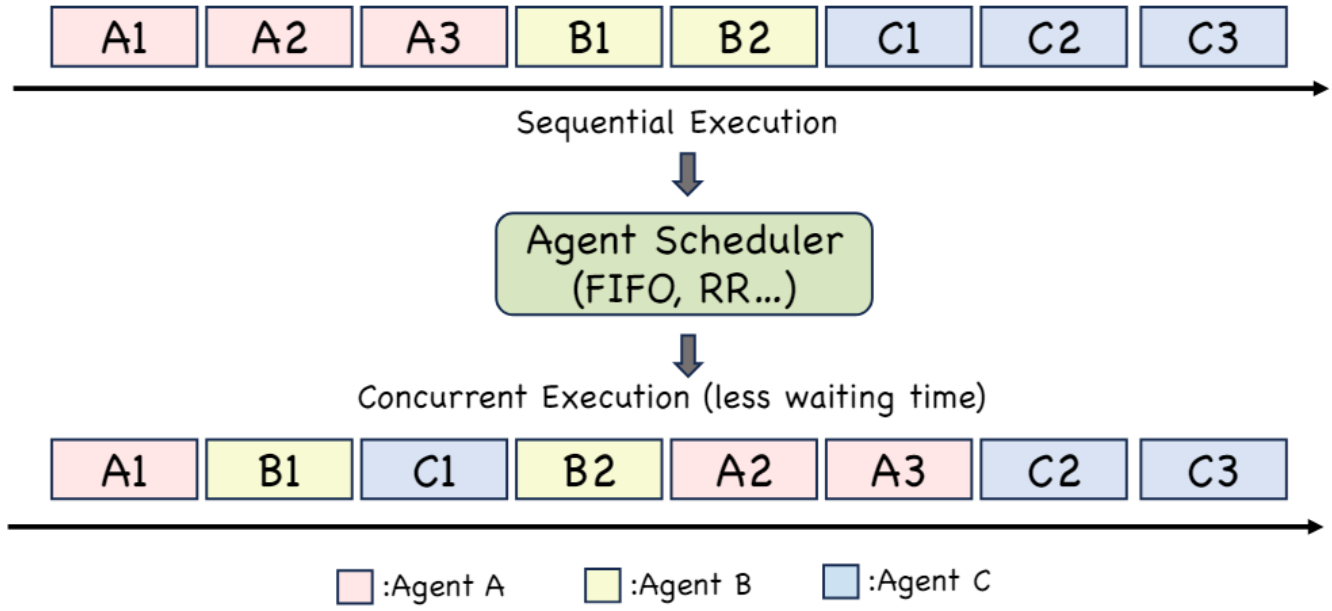

Figure 3: An illustration of the agent scheduler.

假设我们有三个 agent，原本 LLM 在响应 agent 的请求时是线性响应的，如上图所示，LLM 会先处理完 A 的请求，然后处理 B，然后处理 C，但是不同 agent 的请求所需的完成时间可能差距极大，导致一些有时限需求的 agent 饿死，因此引入 agent 调度器，对于属于不同 agent 的请求进行调度，以交错的方式进行处理（如，A1、B1、C1、B2、A2、A3、C2、C3），**确保没有单个智能体垄断处理资源，并且最大限度减少空闲时间。**

可能的方向：

1. 改进智能体调度器的调度算法，优化动态优先级调度；
2. 设计智能体调度器的接口，使得智能体可以自主选择调度策略；
3. 扩展现有调度类的分组逻辑、监控等待时间过长的 agent 并临时提升优先级、可视化调度工具；
4. ...

---

2. **内存管理器 / 存储管理器**：

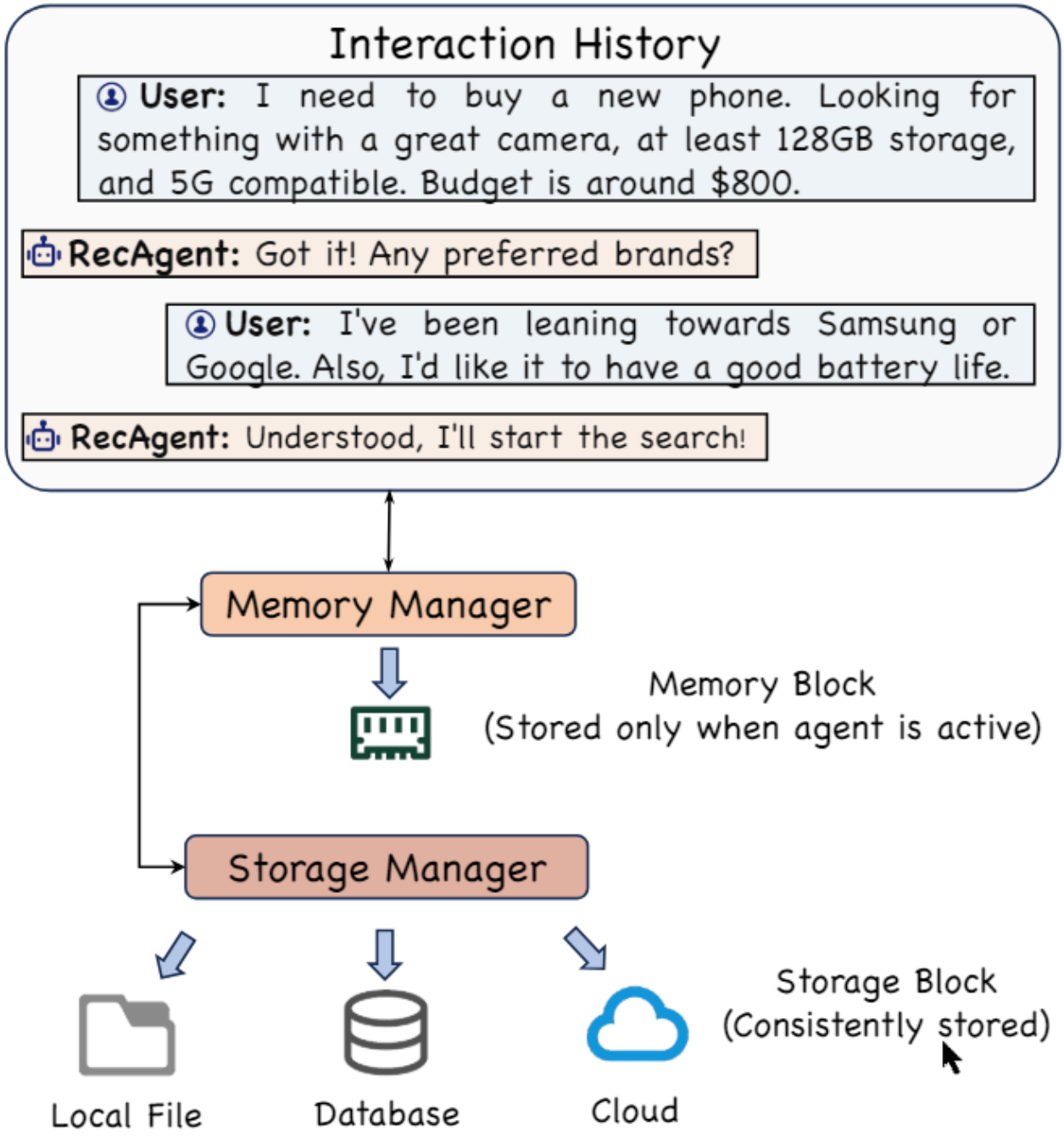

Figure 5: Storage of interaction history with memory manager and storage manager.

内存管理器负责管理智能体的生命周期内的短期内存，确保数据仅在智能体处于活动状态（等待执行或运行时）时，才能存储和访问数据。简单地说，如果我们把每个 agent 看成一个不同的应用，那么这里实现的就是一个应用数据隔离机制。

当前的 AIOS 支持独立存储每个智能体的内存，其他智能体经过访问管理器的授权才能访问每个智能体的内存。内存管理器能够实现快速的数据检索和处理，便于快速响应用户查询和交互。

存储管理器负责数据的长期保存。在 AIOS 中，通过各种持久性介质（如本地文件、数据库或云存储）实现数据的永久存储，以确保数据的完整性和可用性。

可能的方向：

1. ...

---

3. ...

四. 往年选题

1. LLM 嵌入式文件系统：https://github.com/OSH-2024/ArkFS
2. 大模型操作系统助理：https://github.com/OSH-2024/ModelSynergy

五. 参考资料

1. AIOS: AI Agent Operating System：https://github.com/agiresearch/AIOS