Final Project Progress Report

Tao Tong (tt3010)
Kuangyu Chen (kc3827)
Jiacheng Gu (jg4874)

1. Overview

This project transforms the OpenManus LLM agent framework into a distributed, collaborative platform in which specialized agent roles communicate and share context via a message-driven knowledge bus. Beyond throughput gains, the platform enables dynamic role assignment, fine-grained task decomposition, and cross-agent context sharing, mimicking a software engineering team's workflow. We will implement retrieval, planning, execution, validation, and integration agents across multiple nodes, evaluate functionality and performance on GitHub Issue automation tasks, and compare against single-node OpenManus and sequential multi-step baselines. Contributions include a novel multi-role coordination architecture, a shared knowledge bus abstraction, and empirical evidence of improved task accuracy, consistency, and scalability.

We have completed the implementation of retrieval, execution, validation agent, and the message communication bus based on Kafka between them. However, we have encountered many detailed bugs that need to be solved, such as the problem of automatic mock data in the execution agent. Our next step will be to continue to implement planning and integration agents and merge their functions for overall testing and tuning.

2. Research Questions

1) How much does the distributed, multi-agent platform increase task throughput on GitHub Issue automation compared to single-node OpenManus and a sequential multi-step baseline?

2) Does enabling dynamic assignment of agent roles reduce end-to-end latency for completing an Issue automation task?

3) How does fine-grained task decomposition affect overall system performance versus coarser decomposition?

4) To what extent does sharing intermediate context over the knowledge bus

improve the accuracy and consistency of final outputs compared to agents working in isolation?

5) How does increasing the number of compute nodes and corresponding agents impact throughput and resource utilization? Is there a point of diminishing returns?

6) What is the communication overhead introduced by the message-driven knowledge bus, and how does it trade off against the gains from distributed coordination?

3. Value to User Community

This project is mainly designed for software developers. By imitating the role allocation of software development in real technology companies, it designs roles such as product manager, R&D, and testing, which can efficiently carry out complete project development. Whether it is a programmer who thinks that single-point agent development is too slow, or a programmer who feels that he is not competent for large-scale system development due to the limitation of LLM memory window, our project can help them.

Our project adopts a distributed framework, and multi-role multi-node parallel processing of subtasks can improve the throughput and response speed of automated tasks. Because the project has task decomposition, cross-agent context sharing, and automatic evaluation capabilities, it can improve the effectiveness of the project and greatly reduce the workload of manual tuning. In addition, developers can also carry out secondary development of our project according to their own needs and expand more agent nodes.

4. Demo

In the demonstration, we plan to start multiple servers, each server starts one or more agents. Then manually input our business requirements to the planning agent (due to time constraints, we initially decided to implement a simple web application with front-end and back-end), and then display the project implementation progress and the dialogue between agents on the front-end page of our system. When all steps are completed, we run and display the generated application according to the system-generated documents to verify the reliability of our system.

5. Delivery

GitHub: https://github.com/Tongs2000/Multi-Role-LLM-Agent-Platform.git