Revised Final Project Proposal

A Collaborative Multi-Role LLM Agent Platform

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

Abstract

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 100 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.

Background and Motivation

Existing LLM agent frameworks perform tasks monolithically or in fixed sequences, limiting parallelism, modularity, and contextual continuity. Complex software tasks-like triaging issues, generating patches, reviewing code, and merging changes-benefit from distributed collaboration: separate agents specialized for sub-tasks can operate concurrently while sharing intermediate artifacts. Our platform extends OpenManus to support dynamic assignment of roles, incremental knowledge aggregation, and multi-node fault tolerance, advancing beyond performance optimization to richer agent interactions.

Research Objectives

1. Functional Collaboration: Can role-specific agents collaboratively complete multi-step tasks with $\geqslant$30% higher accuracy and consistency than monolithic agents?
2. Context Sharing: Does a shared knowledge bus improve inter-agent communication, reducing redundant calls and error propagation by $\geqslant$40%?
3. Scalable Coordination: How efficiently can tasks be parallelized across $1 \rightarrow 10$ heterogeneous agents (horizontal efficiency $\geqslant$80%)?

System Design

We will design a central Knowledge Bus (Kafka) to route messages and shared context (JSON objects) between role-specific agents deployed in Docker containers. And a coordinator service will assigns tasks based on capability and workload.

We split the overall job into at least five specialized agents:

Retrieval Agent: Gathers all the relevant documentation, code examples, and issue details needed to solve a problem.

Planner Agent: Breaks a big task into smaller, manageable steps so each agent knows exactly what to do.

Executor Agent: Writes code changes or patches and runs tests to make sure they work.

Validator Agent: Checks that the code changes actually fix the issue, flags any failures, and adds comments for improvement.

Integrator Agent: Takes approved fixes and merges them back into the project, creating pull requests for review.

The number of nodes for each agent will depend on workload. All task information and results are saved in a shared data store (PostgreSQL) so agents can stay in sync and nothing gets lost. And we will test our system on 100 real GitHub issues from the dataset "GitHub Public Pull Request Comments", each requiring a patch and pull request. We'll compare our distributed platform against two baselines: the original single-node version of OpenManus and a sequential agent setup. Key measures will include how often tasks are completed correctly, how quickly results are produced, and how well the system scales as we add more agents.

Resources & Dependencies

1. OpenManus https://github.com/mannaandpoem/OpenManus.git
2. Docker, Kafka, PostgreSQL
3. Cloud VM cluster (2× CPU cores, 16GB RAM each)
4. GitHub Public Pull Request Comments
https://www.kaggle.com/datasets/pelmers/github-public-pull-request-comments/data