# KnowLution: A Multi-Agent Framework for Execution-Oriented Autonomous Operations in Production Supercomputers

Yuqi Li*, Liquan Xiao*, Yongqian Sun†, Xiuhong Tan‡, Aiwen Yu‡, Jinghua Feng‡,
Lei Tao†, Tongqing Zhou*, Yuan Yuan*
* *College of Computer Science and Technology, National University of Defense Technology*
liyq@nscc-tj.cn,{xiaoliquan,zhoutongqing,yuanyuan}@nudt.edu.cn
† *Nankai University*, sunyongqian@nankai.edu.cn, leitao@mail.nankai.edu.cn
‡ *National SuperComputer Center in Tianjin*, {tanxh,yuaw,fengjh}@nscc-tj.cn

***Abstract*—Recent advances in large language models (LLMs) and agent-based systems have created new opportunities for autonomous operations in production supercomputers. However, existing approaches remain inadequate for execution-oriented operational tasks, as they cannot reliably transform operational requests into executable workflows under dynamic runtime environments due to the lack of runtime status grounding and execution guarantees. To address this challenge, we propose KnowLution, a multi-agent framework that integrates workflow routing, state-aware retrieval, and execution-aware tool orchestration. KnowLution routes operational requests into structured workflows, grounds reasoning with real-time system states and operational knowledge, and verifies executability before action execution, enabling reliable closed-loop task handling. We evaluate KnowLution on a production-derived benchmark containing 200 representative operational tasks. Experimental results show that KnowLution achieves an 85% task success rate and a 92% tool execution success rate while maintaining practical response latency, substantially outperforming representative baselines.**

***Index Terms*—supercomputers, multi-agent systems, system reliability, Operations management**

## I. Introduction

Modern supercomputing centers rely on increasingly complex operational workflows, including job management, software maintenance, resource scheduling, fault diagnosis, and system recovery. Unlike conventional conversational applications, these operational tasks are execution-oriented and require reliable interaction with a dynamic production environment. As supercomputers continue to scale toward massive node counts, heterogeneous devices, and increasingly complex scheduling policies, these tasks become tightly coupled across multiple system components. Consequently, reliable operation requires decisions grounded in runtime system states and strict execution constraints, since incorrect actions may propagate across schedulers, storage systems, and affect system availability and user workloads [1].

Recent advances in large language models(LLMs)-based intelligent operations systems have significantly improved semantic understanding, operational knowledge retrieval, and task-level reasoning. However, production supercomputing environments require more than intelligent assistance. Operational requests must be transformed into executable workflows grounded in runtime system states and validated against safety, permission, and dependency conditions before execution.

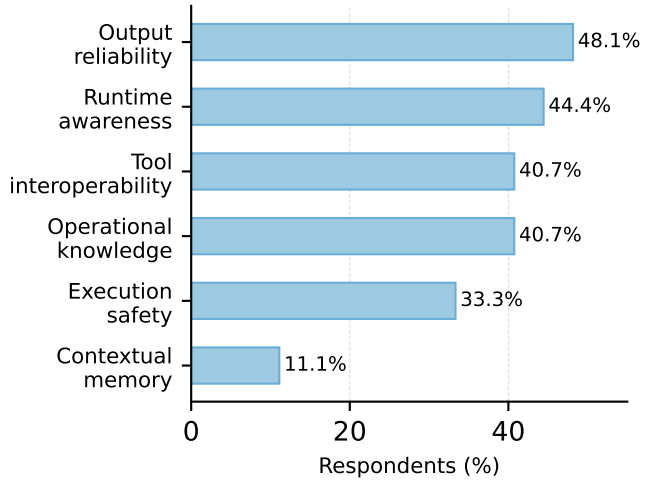


Fig. 1. Perceived limitations of current intelligent operations systems reported by 27 frontline O&M engineers. Multiple choices were allowed.

To better understand the practical requirements, we conducted a questionnaire and interview study involving 27 frontline O&M engineers, over 70% of whom are responsible for production supercomputers. As shown in Fig. 1, respondents identified insufficient output reliability, runtime awareness, tool interoperability, and operational knowledge as major limitations of current intelligent operations systems, while one-third further highlighted execution safety as a key concern. These findings indicate that production deployment requires not only strong reasoning capabilities, but also reliable execution grounded in runtime status, operational knowledge, and execution constraints.

These findings expose a fundamental gap between existing intelligent operations systems and the practical requirements

Yuan Yuan is the corresponding author.

of production supercomputers. Existing approaches fail in three critical aspects. First, retrieval mechanisms mainly rely on static documents or historical cases, making it difficult to align operational knowledge with dynamic runtime status. Second, execution-oriented operational tasks are often treated as conversation queries rather than executable workflows, limiting their ability to coordinate multi-stage operations across heterogeneous system components. Third, although recent agent frameworks support tool invocation, generated actions still lack systematic validation of parameter completeness, dependency conditions, permission compatibility, and execution risks. These limitations are common across existing LLM-based operational assistants [2], [3], RAG-enhanced reasoning methods [4], [5], and general-purpose agent frameworks [6], [7]. Consequently, they cannot reliably transform operational requests into state-grounded and safely executable workflows required by production supercomputers.

To bridge this gap, we propose **KnowLution**, a multi-agent framework for execution-oriented autonomous operations in production supercomputers. By integrating workflow routing, state-aware retrieval, and execution-aware tool orchestration, KnowLution transforms operational requests into state-grounded and safely executable workflows, enabling reliable closed-loop autonomous operations.

The main contributions of this paper are summarized as follows:

- We propose KnowLution, the first execution-oriented multi-agent framework for autonomous operations in production supercomputers. KnowLution transforms operational requests into state-grounded and safely executable workflows through a closed-loop execution process.
- KnowLution integrates workflow routing, state-aware retrieval, and execution-aware tool orchestration to bridge requests, operational knowledge, dynamic runtime status, and safe execution.
- We construct a production-derived benchmark containing 200 representative operational tasks and conduct extensive experiments against representative baselines. KnowLution achieves an 85% task success rate and a 92% tool execution success rate while maintaining engineering-practical response latency.

## II. Related Work

Recent advances in LLM-based intelligent operations systems have significantly improved autonomous operations. However, existing approaches remain insufficient for execution-oriented autonomous operations in supercomputers.

### A. Retrieval-Augmented Knowledge for Intelligent Operations

LLMs that rely solely on parametric memory are prone to outdated knowledge and hallucination, while RAG mitigates these limitations by incorporating external knowledge into model reasoning through a retrieve–augment–generate paradigm [8]. Recent studies improve retrieval from multiple perspectives. Graph- and memory-enhanced methods, such as LightRAG [5] and HippoRAG [9], reorganize fragmented documents into structured memory for improved contextual coverage and multi-hop retrieval. Adaptive-RAG [10] and Self-RAG [4] further optimize retrieval timing and answer–evidence consistency. Agentic RAG [11] and ToolLLM [12] extend retrieval with planning, reflection, and external tool interaction. Despite these advances, existing RAG methods treat retrieved knowledge as static evidence independent of the execution environment. Consequently, they cannot reliably align operational knowledge with dynamic runtime status, motivating state-aware retrieval for execution-oriented operations.

### B. LLM Agents for Autonomous Operations

LLM-based agents extend language models from passive text generators to interactive decision makers capable of reasoning, planning, and tool invocation. Multi-agent collaboration further improves complex task handling through role decomposition and coordinated execution [13]. Representative frameworks, including ReAct [2], AFLOW [14], AutoFlow [15], AutoGen [6], CAMEL [16], MetaGPT [7], and PEER [17], demonstrate strong capabilities in workflow automation, software engineering, and enterprise reasoning. Early O&M-oriented multi-agent systems, such as MAS-CM [18], further illustrate the benefits of hierarchical collaboration for system management.

Recent studies have also explored LLMs and agents for operational scenarios. FlowXpert [3], RCAgent [19], RCA-Copilot [20], mABC [21], and SQLFixAgent [22] support troubleshooting, root-cause analysis, and structured output correction in cloud and database systems. In HPC environments, HPC Coder v2 [23], AutoParLLM [24], LM4HPC [25], and HPC-GPT [26] improve code generation and scientific programming, while HyCE [27], LLM-based workflow anomaly detection [28], chatHPC [29], FoamPilot [30], and LangChain–Parsl [31] assist users with documentation, workflow generation, and job submission.

Nevertheless, existing intelligent operations systems still lack workflow routing and execution-aware tool orchestration, limiting their applicability to execution-oriented operations in production supercomputers.

## III. Design of KnowLution

### A. Overview

Fig. 2 presents the overall architecture of KnowLution. User requests or alerts are first mapped to predefined operational workflows, which are collaboratively executed by specialized agents for system awareness, anomaly analysis, strategy planning, tool orchestration, and result reporting. During execution, state-aware retrieval grounds LLM reasoning with runtime states and operational knowledge, while execution-aware tool orchestration ensures that generated actions satisfy capability and safety constraints. Execution results are further incorporated into the knowledge base, forming a closed-loop autonomous operation framework.

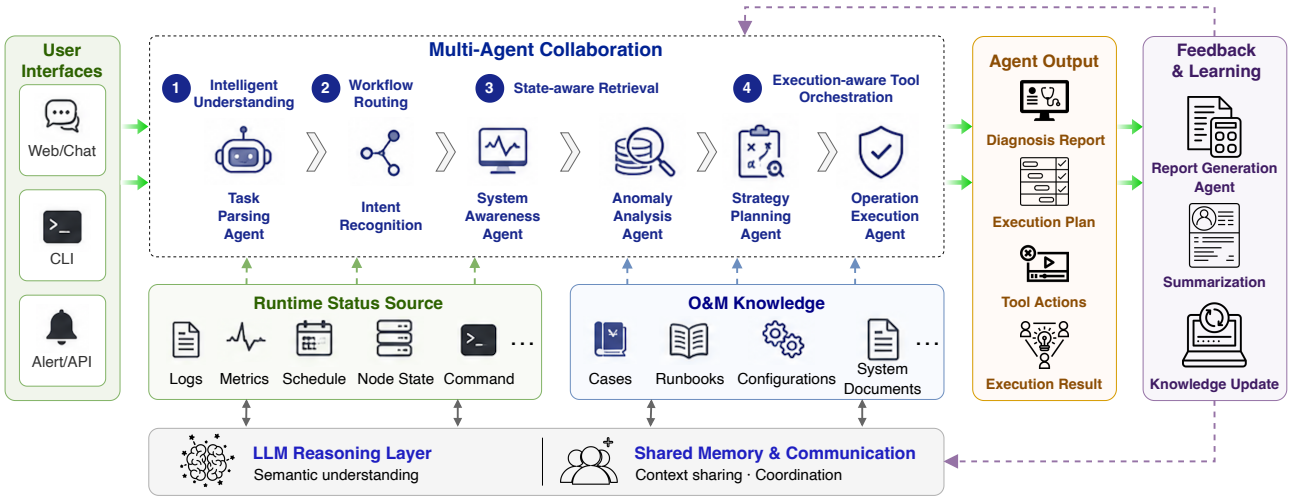


Fig. 2. Architecture of KnowLution.

### B. Workflow Routing

KnowLution treats execution-oriented O&M as *workflow* execution rather than conversational reasoning. Accordingly, user requests are first mapped to predefined operational workflows together with the execution-related entities required for subsequent coordination.

Given a user request $u_t$, the task parsing agent employs an LLM with structured prompts and few-shot demonstrations to identify the workflow type and execution parameters, such as node names, job IDs, or log ranges. The routing process is formulated as:

$$w_t = f_{\text{intent}}\big(\text{LLM}(u_t, \text{ctx}_t)\big), \quad w_t \in \mathcal{W} \cup \{\perp\}, \tag{1}$$

where $\text{ctx}_t$ denotes the contextual representation constructed from historical dialogue and memory, and $\perp$ indicates that the task is beyond the scope that the current system can safely handle.

This resulting workflow defines the execution context for subsequent multi-agent coordination. Lightweight requests invoke only perception and reporting agents, whereas execution-oriented tasks additionally trigger anomaly analysis, planning, and execution, avoiding unnecessary reasoning while improving execution efficiency.

### C. State-aware Retrieval

Operational decisions should be grounded in both dynamic runtime states and accumulated operational knowledge. KnowLution therefore constructs a unified decision context by adaptively weighting three information sources: runtime states, retrieved knowledge, and the semantic representation derived from the current request and interaction context.

Let $\phi_{\text{state}}(\cdot)$, $\phi_{\text{kb}}(\cdot)$, and $\phi_{\text{sem}}(\cdot)$ project the three information sources into a shared semantic space. The resulting context is formulated as:

$$\mathbf{K}_t = \alpha_t \phi_{\text{state}}(s_t) + \beta_t \phi_{\text{kb}}(D_t) + \gamma_t \phi_{\text{sem}}(u_t, c_t), \tag{2}$$

$$(\alpha_t, \beta_t, \gamma_t) = g(w_t), \qquad \alpha_t + \beta_t + \gamma_t = 1, \tag{3}$$

where $s_t$ denotes the current runtime states, $D_t$ represents the retrieved operational documents, and $(u_t, c_t)$ denotes the current request and interaction context. The weighting function $g(\cdot)$ determines the importance of each source according to the routed workflow $w_t$. Runtime-intensive workflows assign greater weight to system states, whereas knowledge-oriented tasks place greater emphasis on retrieved documents.

### D. Execution-aware Tool Orchestration

Tool invocation in production environments requires more than selecting appropriate tools. Practical execution additionally depends on execution order, parameter completeness, dependency satisfaction, permission compatibility, and operational safety. KnowLution therefore generates an execution-aware orchestration plan before tool execution to coordinate tool selection, execution ordering, parameter instantiation, and execution validation under operational constraints.

The strategy-planning agent constructs an executable orchestration plan by jointly optimizing tool selection and execution order while enforcing capability compatibility, parameter completeness, dependency constraints, permission requirements, and execution-risk control. The planning process is formulated as:

$$\begin{aligned} \mathcal{S}^* = \arg\max_{\mathcal{S}} \quad & \sum_{h \in \mathcal{H}} \sum_{(p,\theta) \in \mathcal{S}_h} R(p, \theta \mid q, D_h) \\ \text{s.t.} \quad & 1 \le |\mathcal{S}_h| \le 5, \quad \theta \models \mathcal{C}(p). \end{aligned} \tag{4}$$

The resulting orchestration plan is translated into executable commands or automation playbooks. Write operations are further guarded through sandbox verification and explicit human confirmation, ensuring reliable deployment in production supercomputers.

## IV. Evaluation

This section evaluates KnowLution through the following research questions (RQs):

- **RQ1:** Does KnowLution outperform existing methods?
- **RQ2:** What contributes to KnowLution's performance?

### A. Settings

*1) Dataset:* As summarized in Table I, we constructed a proprietary benchmark comprising 200 anonymized operational tasks derived from real-world production tickets. The benchmark spans three operational dimensions and ten representative task categories, with the category distribution preserved from the source ticket corpus. It includes both routine diagnostic and advisory requests and tasks requiring tool-assisted execution.

*2) Baselines:* We compare KnowLution with five representative baselines covering different paradigms of intelligent operations. Specifically, **LLM-only** employs Qwen3 [32] without external knowledge or tool support. **RAG-only** augments LLM reasoning with operational knowledge. **ReAct** [2] interleaves reasoning with tool invocation. **EasyTool** [33] further supports structured tool selection and parameterized invocation. **MetaGPT** [7] represents role-based multi-agent collaboration with task decomposition and workflow coordination. These methods represent progressively stronger capabilities in knowledge retrieval, tool utilization, and multi-agent collaboration.

*3) Metrics:* We evaluate KnowLution using three metrics: task completion, response quality, and execution reliability.

**Task Success Rate (TSR)** measures the proportion of successfully completed tasks:

$$TSR = \frac{N_{\text{success}}}{N_{\text{total}}}, \tag{5}$$

where $N_{\text{success}}$ and $N_{\text{total}}$ denote the numbers of successful and total tasks, respectively.

**Response Generation Quality** is measured by BLEU-4:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \tag{6}$$

where $p_n$ is the $n$-gram precision, $BP$ is the brevity penalty, and $w_n$ is the corresponding weight.

**Tool Execution Success Rate (TESR)** evaluates execution reliability:

$$TESR = \frac{N_{\text{valid}}}{N_{\text{attempt}}}, \tag{7}$$

where $N_{\text{valid}}$ and $N_{\text{attempt}}$ denote the numbers of successful and attempted tool executions, respectively.

### B. Overall Performance (RQ1)

Table II compares KnowLution with representative baselines. KnowLution achieves the highest TSR (85%) and BLEU-4 (25.7), outperforming the best baseline (ReAct) by 26 percentage points in TSR. These improvements demonstrate that combining workflow routing, state-aware retrieval, and execution-aware tool orchestration is substantially more effective than using retrieval, tool invocation, or multi-agent collaboration alone.

TABLE I
STATISTICS OF OPERATION TASK TYPES.

| No. | Problem Dimension | Task Category | #Tasks | Proportion in Real Tickets |
|---|---|---|---|---|
| 1 | User Support | Knowledge-based Q&A support | 46 | 23.00% |
| 2 | | Job runtime management | 33 | 16.50% |
| 3 | | Account and permission management | 45 | 22.50% |
| 4 | System Operations | System monitoring query | 3 | 1.50% |
| 5 | | Alert and anomaly handling | 4 | 2.00% |
| 6 | | System software environment configuration | 13 | 6.50% |
| 7 | | System service maintenance | 27 | 13.50% |
| 8 | Resource Management | Node resource maintenance | 12 | 6.00% |
| 9 | | Storage-related handling | 7 | 3.50% |
| 10 | | Resource scheduling update and optimization | 10 | 5.00% |
| **Total** | | | **200** | **100.00%** |

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT BASELINE METHODS.

| Method | TSR (%) | BLEU-4 |
|---|---|---|
| LLM-only | 17% | 3.5 |
| RAG-only | 51% | 5.9 |
| ReAct | 59% | 5.3 |
| EasyTool | 49% | 3.8 |
| MetaGPT | 45% | 5.3 |
| **KnowLution (Ours)** | **85%** | **25.7** |

Fig. 3 shows that KnowLution maintains practical latency despite introducing runtime perception and execution validation. Its average latency (28.47 s) is substantially lower than EasyTool (72.18 s) and remains competitive with MetaGPT (38.47 s), demonstrating a favorable balance between execution quality and deployment efficiency.

Fig. 4 reports the execution reliability. ReAct and EasyTool perform redundant tool invocations (90 and 98 calls for 80 required tasks), achieving success rates of only 60.0% and 46.9%, respectively. In contrast, KnowLution completes 70 out of 76 tool invocations (92.0%), demonstrating that workflow routing and execution-aware tool orchestration effectively reduce unnecessary invocations.

### C. Ablation Study (RQ2)

Table III shows that all three components contribute to KnowLution's performance. Removing Tool Orchestration causes the largest decrease in TSR, from **85%** to **54%**, demonstrating that reliable task completion depends critically on validating and coordinating executable actions. Removing Workflow Routing reduces TSR to 72%, highlighting the importance of mapping operational requests to structured workflows for multi-stage coordination. Removing State-aware Retrieval lowers TSR to 78%, confirming that grounding decisions in runtime states and operational knowledge improves contextual correctness.

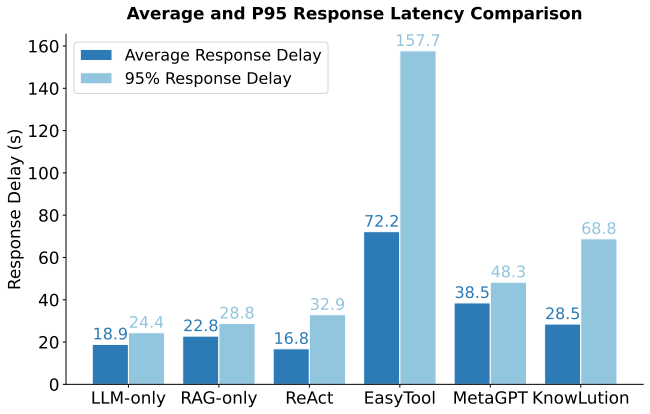


Fig. 3. Comparison of average response latency and 95th-percentile response latency across different methods.

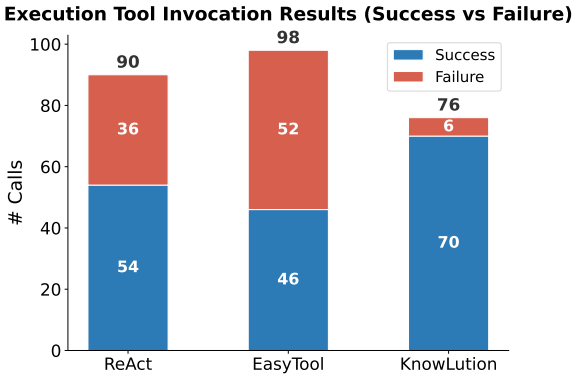


Fig. 4. Tool execution success rate.

All ablated variants also exhibit substantial decreases in BLEU-4, from 25.7% to 5.4–8.5%. Notably, Workflow Routing has the greatest impact on response quality, whereas Tool Orchestration contributes most strongly to task completion. These results demonstrate the complementary roles of the three components: workflow routing structures operational tasks, state-aware retrieval grounds decisions in the current execution context, and tool orchestration translates these decisions into reliable actions.

TABLE III
ABLATION RESULTS OF KNOWLUTION.

| Method | TSR (%) | BLEU-4 (%) |
|---|---|---|
| w/o Workflow Routing | 72 | 5.4 |
| w/o State-aware Retrieval | 78 | 8.2 |
| w/o Tool Orchestration | 54 | 8.5 |
| **KnowLution** | **85** | **25.7** |

## V. DEPLOYMENT EXPERIENCE

During deployment on the Tianhe-NG supercomputer, KnowLution operated entirely within the center's intranet and assisted operators with routine production tasks, while requiring human confirmation for write operations. Post-deployment feedback showed that 81.5% of the participating operators considered its recommendations more reliable, 77.8% reported improved awareness of runtime system states, and 74.1% found that it simplified tool usage. Overall, the feedback suggests that runtime grounding and unified tool access improved support for daily operations and reduced the perceived effort required for troubleshooting.

A typical job-failure handling case further illustrates how KnowLution supports execution-oriented operational tasks in production supercomputing environments. The following cases show how operational requests are transformed into diagnosis and execution workflows through multi-agent collaboration.

*1) Case 1: Job Aborted:* A user reported that an HPC job terminated unexpectedly after submission. KnowLution first collected scheduler information, execution logs, and job status through the system awareness agent. The anomaly analysis agent identified an *out-of-memory* failure caused by insufficient requested memory. Based on the diagnosis, the strategy planning agent recommended increasing memory allocation or MPI parallelism, after which the job completed successfully with the updated configuration.

*2) Case 2: Program Performance Analysis and Optimization:* A second deployment scenario involves a parallel HPC application (miniWeather) exhibiting abnormal execution performance. KnowLution collected runtime logs, scheduler information, and system states, and systematically examined multiple potential bottlenecks. After ruling out CPU/memory utilization and storage quota issues, it identified excessive I/O operations as the primary performance bottleneck. KnowLution then generated optimization recommendations by reducing unnecessary I/O overhead through output configuration optimization, leading to improved application performance.

Together, these deployment results demonstrate that KnowLution can effectively bridge runtime awareness, operational knowledge, and executable decision-making, enabling reliable autonomous assistance for production supercomputing operations.

## VI. CONCLUSION

This paper presented **KnowLution**, a multi-agent framework for execution-oriented autonomous operations in production supercomputers. By combining workflow routing, state-aware retrieval, and execution-aware tool orchestration, KnowLution enables reliable closed-loop execution from natural-language operational requests to safe tool invocation. We constructed a production-derived benchmark containing 200 representative operational tasks and conducted comprehensive evaluations against representative baselines. Experimental results demonstrate that KnowLution consistently improves task success and tool execution while maintaining engineering-practical response latency, validating its effectiveness for production supercomputing environments. In future work, we will investigate self-evolving agent collaboration and adaptive tool orchestration to further improve autonomous operations in dynamic production environments.

## ACKNOWLEDGMENT

This work is supported by the Jing-Jin-Ji Regional Integrated Environmental Improvement-National Science and Technology Major Project under Grant 2025ZD1202200.

## REFERENCES

[1] X. He, X. Chen, H. Guo, X. Liu, D. Chen *et al.*, "Scalability and efficiency challenges for the exascale supercomputing system: practice of a parallel supporting environment on the sunway exascale prototype system," *Frontiers of Information Technology & Electronic Engineering*, vol. 24, pp. 41–58, 01 2023.

[2] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[3] B. Shi, Y. Luo, J. Wang, Y. Zhao, S. Zhang *et al.*, "Flowxpert: Expertizing troubleshooting workflow orchestration with knowledge base and multi-agent coevolution," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, ser. KDD '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 4839–4850.

[4] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *The Twelfth International Conference on Learning Representations*, 2024.

[5] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "LightRAG: Simple and fast retrieval-augmented generation," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 10 746–10 761.

[6] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li *et al.*, "Autogen: Enabling next-gen llm applications via multi-agent conversation," 2023.

[7] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng *et al.*, "MetaGPT: Meta programming for a multi-agent collaborative framework," in *The Twelfth International Conference on Learning Representations*, 2024.

[8] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li *et al.*, "A survey on rag meeting llms: Towards retrieval-augmented large language models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 6491–6501.

[9] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "Hipporag: neurobiologically inspired long-term memory for large language models," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, ser. NIPS '24. Red Hook, NY, USA: Curran Associates Inc., 2024.

[10] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. Park, "Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 7036–7050.

[11] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic retrieval-augmented generation: A survey on agentic rag," *ArXiv*, vol. abs/2501.09136, 2025.

[12] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan *et al.*, "Toolllm: Facilitating large language models to master 16000+ real-world apis," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[13] L. Wang, C. Ma, X. Feng, Z. Zhang, H. ran Yang *et al.*, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, 2023.

[14] J. Zhang, J. Xiang, Z. Yu, F. Teng, X. Chen *et al.*, "Aflow: Automating agentic workflow generation," *arXiv preprint arXiv:2410.10762*, 2024.

[15] Z. Li, S. Xu, K. Mei, W. Hua, B. Rama *et al.*, "Autoflow: Automated workflow generation for large language model agents," *ArXiv*, vol. abs/2407.12821, 2024.

[16] G. Li, H. A. Al Kader Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: communicative agents for "mind" exploration of large language model society," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.

[17] Y. Wang, X. Li, B. Wang, Y. Zhou, Y. Lin *et al.*, "Peer: Expertizing domain-specific tasks with a multi-agent framework and tuning methods," 2024.

[18] D. Grzonka, A. Jakóbik, J. Kołodziej, and S. Pllana, "Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security," *Future Gener. Comput. Syst.*, vol. 86, no. C, p. 1106–1117, Sep. 2018.

[19] Z. Wang, Z. Liu, Y. Zhang, A. Zhong, J. Wang *et al.*, "Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, ser. CIKM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 4966–4974.

[20] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao *et al.*, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems*, ser. EuroSys '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 674–688.

[21] W. Zhang, H. Guo, J. Yang, Z. Tian, Y. Zhang *et al.*, "mABC: Multi-agent blockchain-inspired collaboration for root cause analysis in micro-services architecture," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 4017–4033.

[22] J. Cen, J. Liu, Z. Li, and J. Wang, "Sqlfixagent: towards semantic-accurate text-to-sql parsing via consistency-enhanced multi-agent collaboration," in *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025.

[23] A. Chaturvedi, D. Nichols, S. Singh, and A. Bhatele, "Hpc-coder-v2: Studying code llms across low-resource parallel languages," 2024.

[24] Q. I. Mahmud, A. TehraniJamsaz, H. D. Phan, N. K. Ahmed, and A. Jannesari, "AUTOPARLLM: gnn-guided automatic code parallelization using large language models," *CoRR*, vol. abs/2310.04047, 2023.

[25] L. Chen, P.-H. Lin, T. Vanderbruggen, C. Liao, M. Emani *et al.*, "Lm4hpc: Towards effective language model application in high-performance computing," in *OpenMP: Advanced Task-Based, Device and Compiler Programming: 19th International Workshop on OpenMP, IWOMP 2023, Bristol, UK, September 13–15, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 18–33.

[26] X. Ding, L. Chen, M. Emani, C. Liao, P.-H. Lin *et al.*, "Hpc-gpt: Integrating large language model for high-performance computing," in *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 951–960.

[27] Y. Miyashita, P. K. M. Tung, and J. Barthélemy, "Llm as hpc expert: Extending rag architecture for hpc data," 2024.

[28] H. Jin, G. Papadimitriou, K. Raghavan, P. Zuk, P. Balaprakash *et al.*, "Large language models for anomaly detection in computational workflows: From supervised fine-tuning to in-context learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '24. IEEE Press, 2024.

[29] J. Yin, J. Hines, E. Herron, T. Ghosal, H. Liu *et al.*, "chathpc:empowering hpc users with large language models," *J. Supercomput.*, vol. 81, no. 1, Nov. 2024.

[30] L. Xu, D. Mohaddes, and Y. Wang, "Llm agent for fire dynamics simulations," 2024.

[31] H. Ma, A. Brace, C. Siebenschuh, I. Foster, and A. Ramanathan, "Langchain-parsl: Connect large language model agents to high performance computing resource," in *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC Workshops '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 78–85.

[32] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui *et al.*, "Qwen3 technical report," 2025.

[33] S. Yuan, K. Song, J. Chen, X. Tan, Y. Shen *et al.*, "EASYTOOL: Enhancing LLM-based agents with concise tool instruction," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 951–972.