# InfraGym: Empowering LLM Agents for Real-World Computer System Optimization

**Huaizheng Zhang**[1] **Lei Zhang**[1] **Yuanming Li**[1] **Yizheng Huang**[2] **Xiaotong Yang**[1]
**Kuntai Du**[3] **Yihua Cheng**[3] **Junchen Jiang**[3] **Wencong Xiao**[1]
[1]MLSysOps [2]UCLA [3]University of Chicago

## Abstract

Large language model (LLM) agents have demonstrated high potential of improving performance for complex computer systems, such as cluster scheduling, network congestion control, and adaptive video streaming. However, in the lack of a standard, safe, and extensible benchmarking platform, it is difficult to evaluate whether these LLM agents improve real-world system performance and by how much. We present InfraGym, an open, extensible platform where researchers can study computer system optimization with LLM agents. Our current release includes three real-world cases and supports interaction with both simulated and real environments. We benchmark multiple LLM agents on these tasks using both open-source and closed-source LLMs, and outline future directions. The code is available at https://github.com/MLSysOps/InfraGym

## 1 Introduction

Beyond traditional agentic applications such as game playing [1, 2] and medical assistance [3, 4], we found that Large Language Model (LLM) agents start to demonstrate great potential of optimizing complex real-world computer systems, such as load balancing [5], adaptive video streaming [6] and network congestion control [7, 8]. These systems directly affect billions of users and are critical to technology companies. For example, load balancing distributes requests across servers to balance resource utilization, while video streaming relies on adaptive bitrate control to maintain a consistent user experience under varying bandwidth conditions.

The success of LLM agents in optimizing computer systems is not random. Before LLMs, Reinforcement-Learning-based (RL-based) agents are widely studied by both machine learning and system research literature [9–11]. Those RL agents can achieve high performance in controlled research environments. However, they struggle to generalize to dynamic, real-world systems [12, 13]. Further, their reliance on complex training pipelines and large datasets also limits their adoption. To understand the abilities of these RL agents, RL platforms such as Park [9] were built to standardize training and evaluation of RL agents in computer systems. In contrast, LLM-based agents can generalize to real-world systems without extra training, making them much more robust when optimizing complex real-world computer systems.

However, existing RL platforms are poorly suited for LLM agents. First, they assume structured numeric states and low-level actions (e.g., bitrate levels), while LLM agents rely on natural language descriptions. Second, RL platforms focus more on enabling training through trial-and-error for a specific case, which limits generalization and interpretability. In contrast, LLM agents excel at inference-time adaptation, reasoning [14–16], and in-context learning [17, 18]. Third, RL platforms lack LLM-required flexibility. LLM agents often need preloaded prompts [19], memory [20, 21], or tools such as web search [22] and calculators [23], none of which are supported by current RL platforms. Thus, researchers can not plug-and-play them to benchmark agent performance with ease.

ML For Systems workshop at Neural Information Processing Systems (NeurIPS 2025).

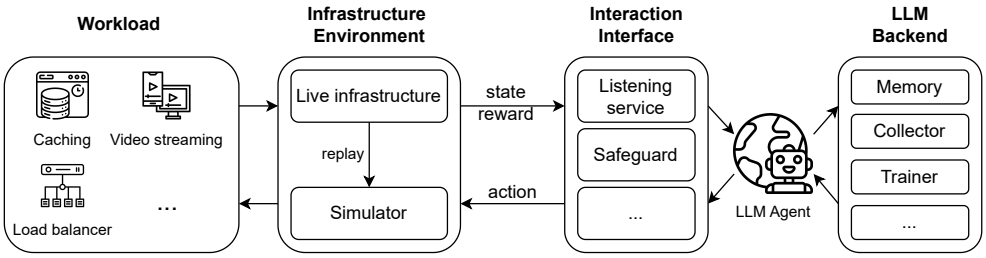



Figure 1: Overview of InfraGym. Infrastructure environments (simulated or live) send states and rewards through the interaction interface to LLM agents. Agents return actions to the interface, which validate actions before sending them to the environments. Moreover, the collector and trainer support trajectory logging and further learning.

To fill the gap, we propose to develop a new platform for building, testing, and improving LLM agents on computer system optimizations. Our design principles are as follows. First, the new system should treat natural language description as first-class input. This improves agent understanding and provides more interpretability for humans. By doing so, we open the black box of system optimizations and enable human-agent co-learning. Second, the system should pay more attention to the inference stage and try to explore LLMs' native abilities in reasoning, long-context processing, and tool use. Third, the platform should be highly customized. Users are able to configure the memory size, tool use, etc, to better benchmark LLM agents' ability under controlled settings.

We propose InfraGym, an open, standardized, and extensible environment for both ML and system communities to study LLM agents on computer systems. First, it supports both natural language and numeric inputs and processes real-world computer systems in different ways. Second, it provides a unified interface for LLM agents to operate on simulated or production environments with ease. Meanwhile, the interface is human-readable for interpretability, validates actions for safety, and follows OpenAI Gym conventions [24] with added plug-and-play flexibility. Finally, it allows to record agent–environment trajectories for later offline analysis and reinforcement fine-tuning.

We evaluate agents powered by three LLMs (GPT4 [25], Gemini [26], and Qwen3 [27]) across three representative environments: video streaming, caching, and load balancing. Results show the great potential of LLM agents in reducing cost and improving resource utilization while also discussing their limitations. We open-sourced InfraGym and plan to expand it with more environments and agents to foster community-driven research: https://github.com/MLSysOps/InfraGym.

## 2 The InfraGym Design

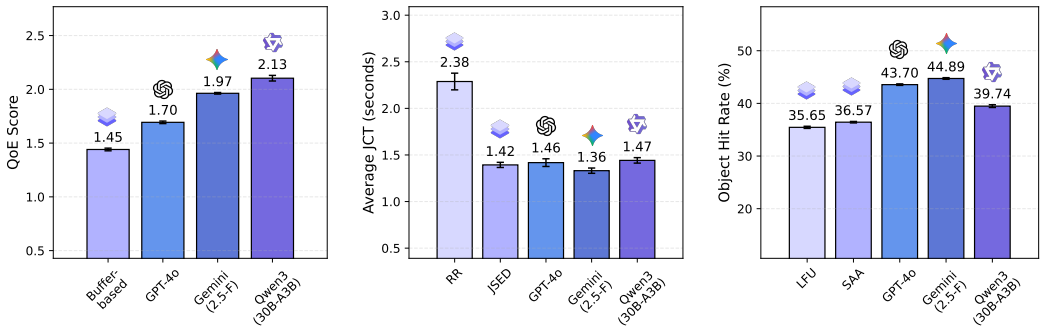
This section describes InfraGym's architecture (Figure 1). Computer environments send states and rewards to LLM agents for decision-making. Between them, we build an interaction interface to translate raw, unstructured data into a structured observation space with natural language descriptions. This translation helps agents interpret environmental changes and plan subsequent actions. The interface also serves as a safeguard by blocking risky [28] or invalid actions [29]. In addition, InfraGym provides a collector and trainer to support continuous learning and adaptation in a real-world environment.

```python
import infragym
from infragym import Agent

# Create a load balancing environment
env = infragym.make('load_balance',
                    num_server=3,
                    max_episode_steps=1000,
                    reward_scale=1.0)
# Init the env
obs, info = env.reset()
# Define the Agent
agent = Agent(obs['task_description'],
              obs['available_actions'])
# Agent-env interaction loop
While True:
  server_index = agent.action(obs['text_observation'])
  obs, reward, done, truncated, info = env.step(1)
# Close the env
env.close()
```

Figure 2: Example Interface of InfraGym

**Infrastructure environment.** InfraGym supports both simulated and live environments, each with its own states, actions, and reward functions. Simulated environments can be used in two ways: replaying real-world traces to reproduce historical events, or generating synthetic data to explore edge cases. Live environments allow direct interaction with production systems. To integrate with them, users need to follow our standards and examples to define their own wrappers.




(a) Video streaming QoE (Higher the better)

(b) Load balancing average JCT (Lower the better)

(c) Caching hit rate (Higher the better)

Figure 3: Benchmark results on InfraGym. LLM agents outperform rule-based baselines in video streaming, load balancing, and caching, showing both effectiveness and variability across LLMs.

**Interaction interface.** This component serves two purposes. First, it receives raw RPC requests (states and rewards) from environments and converts them into structured observations [30]. Users may choose to attach natural language descriptions as shown in Figure 2, which improves agent comprehension, or omit them to test agents' ability to interpret raw data and reduce inference latency. Second, the interface validates actions from agents and returns executable commands via RPC. This design ensures safe, seamless integration of diverse environments and agents.

**LLM backend.** The backend includes three components: agents, a collector, and a trainer. Users can implement agents with different LLMs using a unified template, and our backend supports fast prototyping of prompts and workflows. Specifically, we adopt LangGraph [31] to build the agent template. Meanwhile, the collector, built with Langfuse [32], records trajectories of agent–environment interactions for analysis and future training. Finally, a trainer wrapper built on the Verl framework [33] supports reinforcement learning to further improve agent performance [34].

## 3 Benchmark Studies

We evaluate InfraGym on three representative computer system optimization tasks: adaptive bitrate (ABR) streaming, caching, and load balancing. The reason we choose them is that they all reflect core challenges in computer system optimization, though their detailed targets are different (e.g., ABR for user experiences and caching for lower latency).

**Experimental settings.** We benchmark a rule-based baseline and three LLM agents including GPT-4o [25], Gemini-2.5-Flash [26], and Qwen3-30B (without reasoning mode) [27]. Each experiment is repeated ten times with different seeds. For the ABR and load balancing tasks, we use simulated environments with 50 warm-up steps followed by 200 test steps. For the caching task, we use a real-world trace with 200 warm-up steps and 1000 test steps.

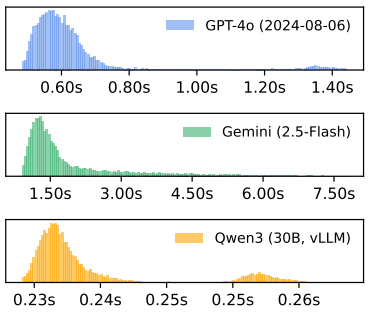



Figure 4: LLM agent latency distributions

**Effectiveness.** Figure 3 shows, LLM agents consistently outperform rule-based baselines across all three tasks. In video streaming, they deliver higher quality-of-experience scores, with Qwen3-30B achieving the best result (2.13) compared to the buffer-based baseline (1.45). We hypothesize that this advantage comes from the model's smaller size: while larger models sometimes "overthink" and hallucinate in simple decision tasks, Qwen3-30B produces more direct and stable responses, which is beneficial for ABR control. In load balancing, all LLMs reduce average job completion time, with Gemini-2.5-Flash performing best at 1.36 s versus 2.38 s for round robin. In caching, LLM agents also surpass rule-based methods such as Least Frequently Used (LFU) and Size-Aware Admission (SAA), with

Gemini-2.5-Flash reaching a 44.9% hit rate compared to 35–37% for rules. These results confirm that LLM agents can generalize across different infrastructure problems, though the performance gap among models suggests opportunities to further improve robustness and efficiency.

**Efficiency.** Figure 4 shows the latency distribution of the three LLMs when applied to load balancing. Qwen3-30B, deployed with vLLM, achieves the lowest and most stable latency (0.23–0.26 s). GPT-4o exhibits moderate latency (0.6–1.4 s), striking a balance between responsiveness and reasoning quality. Gemini-2.5-Flash, while achieving high effectiveness, incurs the highest latency (1.5–7.5 s), limiting its suitability for latency-sensitive environments. These results reveal a trade-off between capability and efficiency, pointing to future work on hybrid strategies that combine fast small models with powerful large ones.

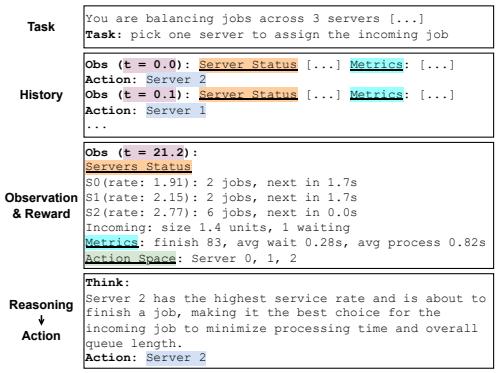



Figure 5: InfraGym case study

**Cast study.** Beyond performance, InfraGym enables interpretability by exposing how LLM agents reason and act. Figure 5 shows a load balancing example, where the agent explains its choice of sending a request to a server in natural language. Unlike RL agents, which often act as black boxes, LLM agents provide step-by-step rationales that help researchers debug and refine their behavior. As our InfraGym's supports this human-in-the-loop manner, users can understand agent mistakes, and then adjust prompts to improve agent policies iteratively.

## 4 Related Work

We review two lines of related work: LLM agents for computer system optimization, and gym-style environments for benchmarking.

**LLM agents for computer system.** Recent studies applied LLM agents to computer systems. One line of work [5] uses the ReAct framework for multi-objective HPC job scheduling, where LLM agents improved throughput, wait time, and fairness over FIFO and Google OR-Tools by leveraging adaptive reasoning. Other work [7, 8] explored congestion control with LLM agents, showing promising results, while NetLLM [6] unified several network optimization tasks under a single LLM framework. Unlike these task-specific efforts, InfraGym provides an open, extensible platform for both LLM and infrastructure researchers to benchmark and improve agents across diverse optimization problems.

**Gym environments.** Open gym environments are crucial for benchmarking and refining LLM agents. ML-Gym [35] and ML-Dojo [36] standardize machine learning tasks for evaluating ML engineering agents. SWE-Gym [37] builds software development environments for training software engineering agents. AgentGym [38] extends evaluation to interactive domains such as the web and games. In contrast, InfraGym focuses on low-level computer systems, where we need specialized LLM agents to deal with distinct challenges.

## 5 Conclusion

We presented InfraGym, an open platform to study LLM agents for computer system optimization. InfraGym provides three key features: a connector for both simulated and live environments, a flexible LLM backend, and a unified interaction interface. We benchmark multiple LLM agents on three representative tasks, and the results highlight a promising direction. Despite encouraging results, InfraGym also reveals challenges in the use of LLM agents. For example, Large models often incur high inference costs and response delays, and their decisions can be inconsistent. Looking ahead, we will extend InfraGym with more environments and agents, and explore hybrid strategies that combine fast small models with powerful large ones. We invite both ML and system communities to try our open-source release https://github.com/MLSysOps/InfraGym.

## References


[1] Sihao Hu, Tiansheng Huang, Gaowen Liu, Ramana Rao Kompella, Fatih Ilhan, Selim Furkan Tekin, Yichang Xu, Zachary Yahn, and Ling Liu. A survey on large language model-based game agents. *arXiv preprint arXiv:2404.02039*, 2024.

[2] Jen-tse Huang, Eric John Li, Man Ho Lam, Tian Liang, Wenxuan Wang, Youliang Yuan, Wenxiang Jiao, Xing Wang, Zhaopeng Tu, and Michael Lyu. Competing large language models in multi-agent gaming environments. In *The Thirteenth International Conference on Learning Representations*, 2025.

[3] Wenxuan Wang, Zizhan Ma, Zheng Wang, Chenghan Wu, Jiaming Ji, Wenting Chen, Xiang Li, and Yixuan Yuan. A survey of llm-based agents in medicine: How far are we from baymax? *arXiv preprint arXiv:2502.11211*, 2025.

[4] Xi Chen, Huahui Yi, Mingke You, WeiZhi Liu, Li Wang, Hairui Li, Xue Zhang, Yingman Guo, Lei Fan, Gang Chen, et al. Enhancing diagnostic capability with multi-agents conversational large language models. *NPJ digital medicine*, 8(1):159, 2025.

[5] Prachi Jadhav, Hongwei Jin, Ewa Deelman, and Prasanna Balaprakash. Evaluating the efficacy of llm-based reasoning for multiobjective hpc job scheduling. *arXiv preprint arXiv:2506.02025*, 2025.

[6] Duo Wu, Xianda Wang, Yaqi Qiao, Zhi Wang, Junchen Jiang, Shuguang Cui, and Fangxin Wang. Netllm: Adapting large language models for networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 661–678, 2024.

[7] Deol Satish, Shiva Raj Pokhrel, Jonathan Kua, and Anwar Walid. Distilling large language models for network active queue management. *arXiv preprint arXiv:2501.16734*, 2025.

[8] Shyam Kumar Shrestha, Shiva Raj Pokhrel, and Jonathan Kua. Adapting large language models for improving tcp fairness over wifi. *arXiv preprint arXiv:2412.18200*, 2024.

[9] Hongzi Mao, Parimarjan Negi, Akshay Narayan, Hanrui Wang, Jiacheng Yang, Haonan Wang, Ryan Marcus, Mehrdad Khani Shirkoohi, Songtao He, Vikram Nathan, et al. Park: An open platform for learning-augmented computer systems. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 197–210, 2017.

[11] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.

[12] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 495–511, 2020.

[13] Fernando Martinez-Lopez, Tao Li, Yingdong Lu, and Juntao Chen. In-context reinforcement learning via communicative world models. *arXiv preprint arXiv:2508.06659*, 2025.

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[15] Shunyu Yao, Jeffrey Zhao, D Yu, N Du, I Shafran, K Narasimhan, and Y Cao. React: Synergizing reasoning and acting in language models. doi: 10.48550. *arXiv preprint ARXIV.2210.03629*, 2022.

[16] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

[17] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[18] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[19] Mathurin Videau, Alessandro Leite, Marc Schoenauer, and Olivier Teytaud. Evolutionary pre-prompt optimization for mathematical reasoning. *arXiv preprint arXiv:2412.04291*, 2024.

[20] Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *ACM Transactions on Information Systems*, 2024.

[21] Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[22] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

[23] Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37: 75749–75790, 2024.

[24] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. 2016. eprint. *arXiv preprint arXiv:1606.01540*, 50, 2016.

[25] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[26] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[27] A Yang Qwen, Baosong Yang, B Zhang, B Hui, B Zheng, B Yu, Chengpeng Li, D Liu, F Huang, H Wei, et al. Qwen2. 5 technical report. *arXiv preprint*, 2024.

[28] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[29] Yixin Dong, Charlie F Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024.

[30] Andrew D Birrell and Bruce Jay Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, 2(1):39–59, 1984.

[31] LangGraph Team. Langgraph — build resilient language agents as graphs., 2024. URL `https://github.com/langchain-ai/langgraph`. Software available from https://github.com/langchain-ai/langgraph.

[32] Clemens Rawert, Marc Klingen, and Maximilian Deichmann. Langfuse — open-source llm engineering platform, 2023. URL `https://langfuse.com/`. Software available from https://langfuse.com.

[33] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

[34] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

[35] Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

[36] Rushi Qiang, Yuchen Zhuang, Yinghao Li, Rongzhi Zhang, Changhao Li, Ian Shu-Hei Wong, Sherry Yang, Percy Liang, Chao Zhang, Bo Dai, et al. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering. *arXiv preprint arXiv:2505.07782*, 2025.

[37] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

[38] Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.