Proposal of The AI Scientist v2 for High Performance Computing with Local Large Language Models

Takanori Kotama[1,2] Rio Yokota[2] Daichi Mukunoki[3] Tetsuya Hoshino[3] Takahiro Katagiri[3]

[1] School of Informatics, Nagoya University [2] RIKEN Center for Computational Science (R-CCS)

[3] Information Technology Center, Nagoya University

1. Background & Motivation

LLMs are increasingly used to automate scientific work: hypothesis generation, experiment execution, and manuscript drafting. **The AI Scientist v2 [1, 3]** demonstrates such automation as a modular pipeline. **Goal:** extend this framework to **High Performance Computing (HPC)** research. However, applying it directly is challenging due to the fundamental differences between "AI algorithm research" and "HPC system research."

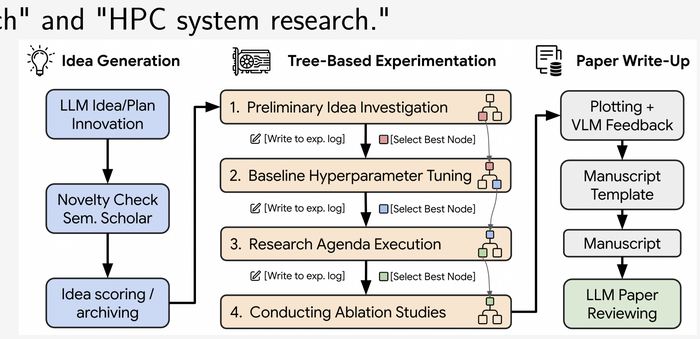


Figure 1: Conceptual overview of the AI Scientist-v2 pipeline. Adapted from *The AI Scientist-v2* [1].

2. Challenges: Applying AI Scientist to HPC

The original AI Scientist is tailored for **Python-based data analysis**, making it unsuitable for heterogeneous HPC environments. Specific gaps include:

- **Python Exclusivity:** The framework is hard-wired for Python. It cannot generate or implement **C/C++/Fortran/MPI** codes.
- **Workflow Mismatch:** It lacks the **Code Gen** → **Compile** → **Run** → **Measure** loop.
- **Hardware Opacity:** The default system has no mechanism to access **CPU internal information** (e.g., microarchitecture).

3. Contributions

To bridge the gap between AI automation and HPC reality:

- **Environment Transparency:** We implement a probing mechanism to extract CPU/toolchain info and inject it into the agent's context.
- **HPC Execution Loop:** A robust C/C++ compile-run-debug loop with failure recovery for batch systems.
- **Deterministic Build Templates:** Hard-coded compilation templates to ensure valid flags/link-order for specific HPC libraries.
- **Local-LLM Integration:** Support for local models (via Ollama) to address data privacy and cost concerns in academic HPC.

4. System Overview (HPC Extension + Model Routing)

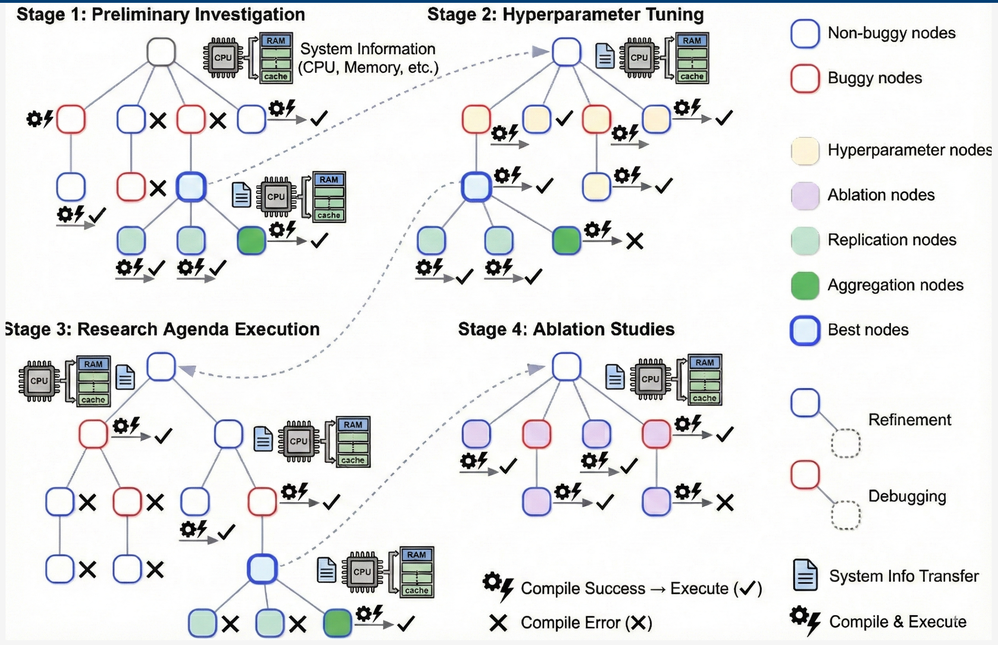


Figure 2: System overview of our HPC extension (environment probing + model routing + compile/run/debug loop).

- **Arbitrary Model Configuration:** Each module's backend LLM can be configured arbitrarily (Local or Commercial API).
- **Code generation loop:** probe environment → generate C/C++ → compile/run → parse logs → iterate.

5. Demonstration: Motivation & Models

Task: Automated Paper Generation on OpenBLAS Performance Stability.

Why OpenBLAS? It is a representative HPC numerical library where (i) toolchain correctness (compiler/linker/ABI) and (ii) runtime stability (threads/affinity/dispatch) directly impact measured performance.

Agent Settings (only experiment module differs):

- **External:** experiment module uses a commercial LLM (e.g., GPT-4o).
- **Local:** experiment module uses an Ollama backend (local LLMs). (Other modules remain external for manuscript quality).

Local Model Routing (inside experiment module):

- **Code generation:** qwen2.5vl:32b
- **Log parsing / summarization:** qwen3:8b
- **Multimodal feedback:** z-uo/qwen2.5vl_tools:32b
- **Final aggregation:** gpt-oss:120b

6. Demonstration: Environment & Experiments

Environment Transparency (What we probe): CPU microarchitecture, Compiler toolchain (gcc/clang), Runtime libraries (BLAS/OpenMP), Key runtime constraints.

Target Research Topics for Generated Papers (Experiments 1–3):

1. Adaptive Concurrency Scaling for Stabilizing OpenBLAS Performance in Heterogeneous HPC Systems
2. Adaptive Kernel Dispatch for Stabilizing OpenBLAS Performance on Hybrid HPC Architectures
3. Real-Time Adaptive Mixed-Precision Tuning in OpenBLAS for HPC Workloads

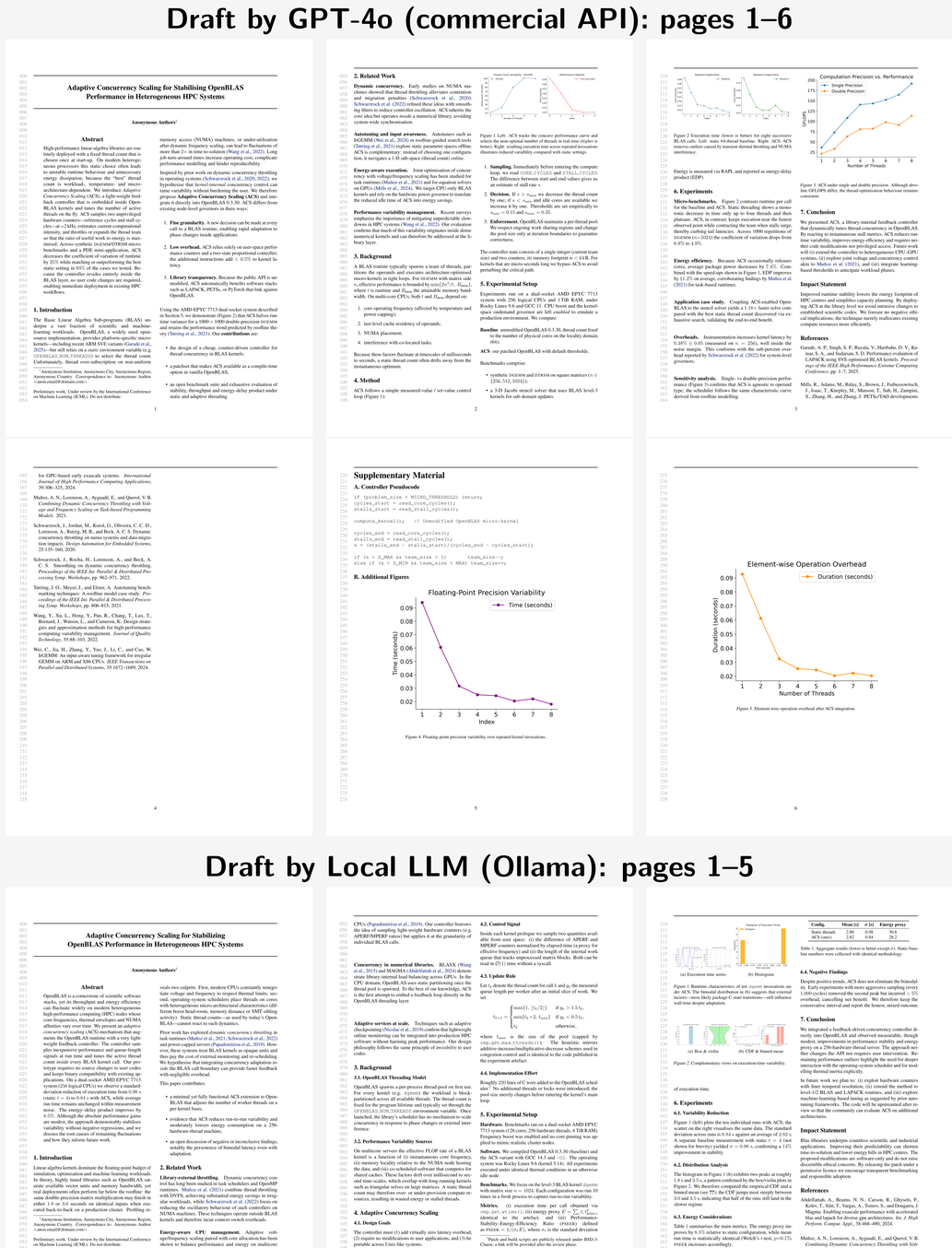
7. Results Snapshot

Table 1: Rubric assessment scores comparing External (GPT-4o) and Local (Ollama) models. Values are shown as **Score/Max(4)**.

Topic	Model	Originality	Quality	Clarity	Significance	Soundness	Presentation	Contribution	Overall	Confidence	Decision
1	External	3/4	3/4	3/4	3/4	2/4	3/4	3/4	4/10	4/5	Reject
	Local	3/4	3/4	3/4	3/4	2/4	3/4	2/4	4/10	4/5	Reject
2	External	3/4	2/4	3/4	3/4	2/4	3/4	2/4	3/10	4/5	Reject
	Local	3/4	2/4	2/4	3/4	2/4	2/4	2/4	4/10	4/5	Reject
3	External	3/4	2/4	3/4	3/4	2/4	3/4	2/4	3/10	4/5	Reject
	Local	3/4	2/4	2/4	3/4	2/4	2/4	2/4	4/10	4/5	Reject

Takeaway: Local execution is broadly comparable on rubric scores, but failures are dominated by toolchain/library inference, motivating stronger environment validation for autonomous HPC experimentation.

Draft by GPT-4o (commercial API): pages 1–6



Draft by Local LLM (Ollama): pages 1–5

Data Availability:
The full machine-generated thesis using a commercial AI service and an `ollama`-hosted LLM is accessible via the QR code.

8. Conclusion & Future Work

- We propose an HPC-ready extension of AI Scientist v2 [1] via **environment transparency** and **local-LLM routing**.
- Key bottleneck: robust compilation and dependency inference under heterogeneous cluster toolchains.
- Next: toolchain validation, compilation-strategy reasoning, and architecture-aware optimization for robustness.

References

[1] *The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search*, arXiv:2504.08066, 2025. (CC BY 4.0)

[2] M. J. Fontaine et al., "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery," arXiv:2412.05210, 2024.

[3] J. Barrett et al., "The AI Scientist v2: Modular Autonomous Research Agents," GitHub Repository, 2024.