HPC-AutoResearch: Adapting Autonomous Research Systems for HPC through Split-Phase Execution

Split-Phase Execution & Compressed Inference Memory on a Best-First Tree Search

Takanori Kotama[1,2], Shun-ichiro Hayashi[1], Tetsuya Hoshino[1],
Daichi Mukunoki[1], Satoshi Ohshima[3], Rio Yokota[2], Takahiro Katagiri[1]

[1]Nagoya University, Aichi, Japan [2]RIKEN Center for Computational Science, Hyogo, Japan
[3]Kyushu University, Fukuoka, Japan

kotama@hpc.itc.nagoya-u.ac.jp

Japan–France Joint Workshop on LLM-Driven Code Generation
and Automation for HPC and Scientific Computing
Sorbonne University (LIP6), Paris — July 6, 2026

Outline

1. Background & Challenges: HPC for autonomous research
2. Contributions
3. Proposed Method
   - Split-Phase Execution Model *(primary)*
   - Compressed Inference Memory *(supporting)*
4. Experiments & Results
   - Comparison vs. general-purpose CLI agents
   - Memory ablation & cross-stage robustness
5. Discussion, Conclusion & Future Work

Background: Autonomous Research, but Only for Python ML

LLM-based autonomous research systems

AI Scientist[1] runs the **full scientific cycle** — hypothesis → experiments → paper writing → peer review — **without human intervention**.

The gap

All existing systems are designed **exclusively for Python-based ML** experiments.
An entire class of scientific computing is left untouched: **High-Performance Computing**.

HPC means compiled languages, batch-scheduler clusters (SLURM), and iterative tuning campaigns
— **none of which current systems support**.

[1] AI Scientist (Lu et al., arXiv:2408.06292, 2024) introduced the full autonomous cycle; we build on **AI Scientist v2** (Yamada et al., arXiv:2504.08066, 2025), which explores candidate experiments as a *Best-First tree search*.

Three Fundamental Gaps When Applying AI Scientist v2 to HPC

1. **Multi-stage failure propagation**
   One bad env setting or compile error at *any* stage aborts the whole pipeline — **no autonomous repair**.
2. **Rootless execution requirement**
   Shared clusters ban Docker's daemon model ⇒ **Singularity/Apptainer** for isolation.
3. **Finite context window**
   Knowledge across *hundreds* of debugging steps **exceeds any LLM's context**.

(+) ML-specific review prompts cannot fairly evaluate HPC research.

HPC-AutoResearch — the **first** autonomous research system for HPC,
via **Split-Phase Execution** and **Compressed Inference Memory**.

Positioning vs. Prior Autonomous Research Systems

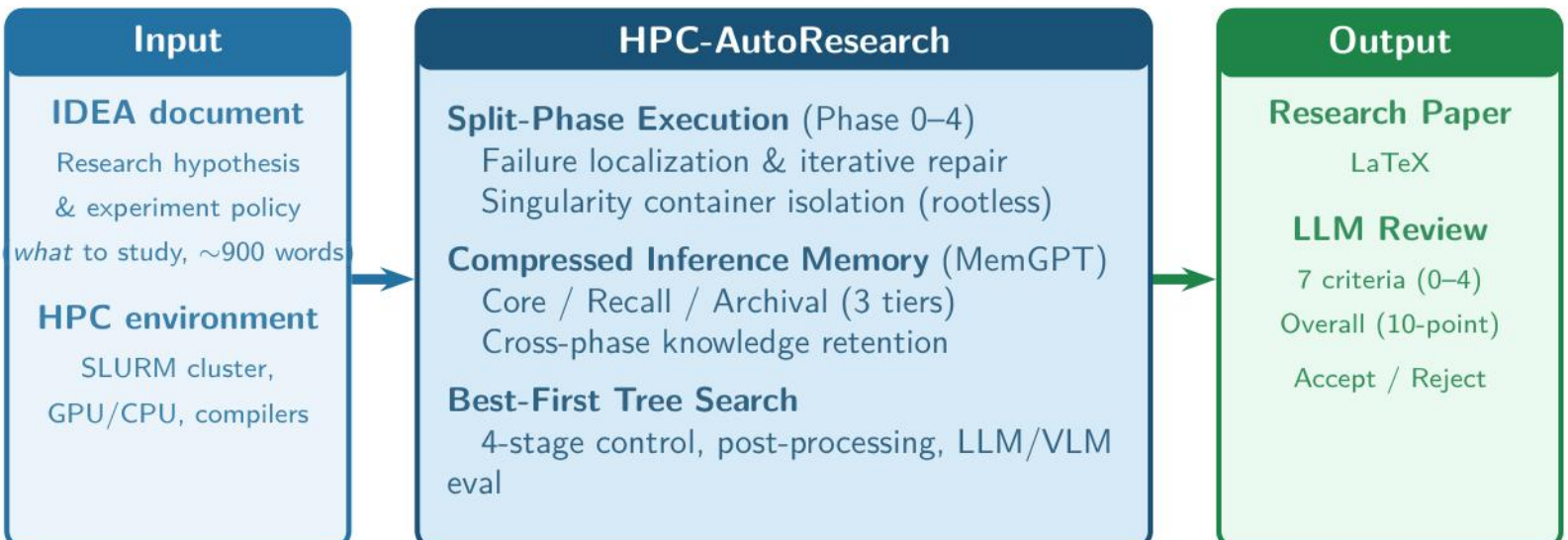
	AI Scientist (Lu et al., 2024)	**AI Scientist v2** (Yamada et al., 2025)	**HPC-AutoResearch** (This work)
Target domain	ML	ML	**HPC / general**
Languages	Python	Python	**C/C++/Fortran/CUDA**
Execution model	Linear	Tree search	**Split-Phase + tree**
Env isolation	None	None	**Singularity (rootless)**
Dep. resolution	Pre-installed	Pre-installed	**Iterative ReAct**
Long-term memory	None	Parent node only	**3-tier (Core/Recall/Arch.)**
HPC env. detection	None	None	**Auto-detect**
Result validation	None	LLM (ML-specific)	**LLM (generalised)**

Highlighted = new or extended in this work — integrates all capabilities needed for HPC.

Contributions

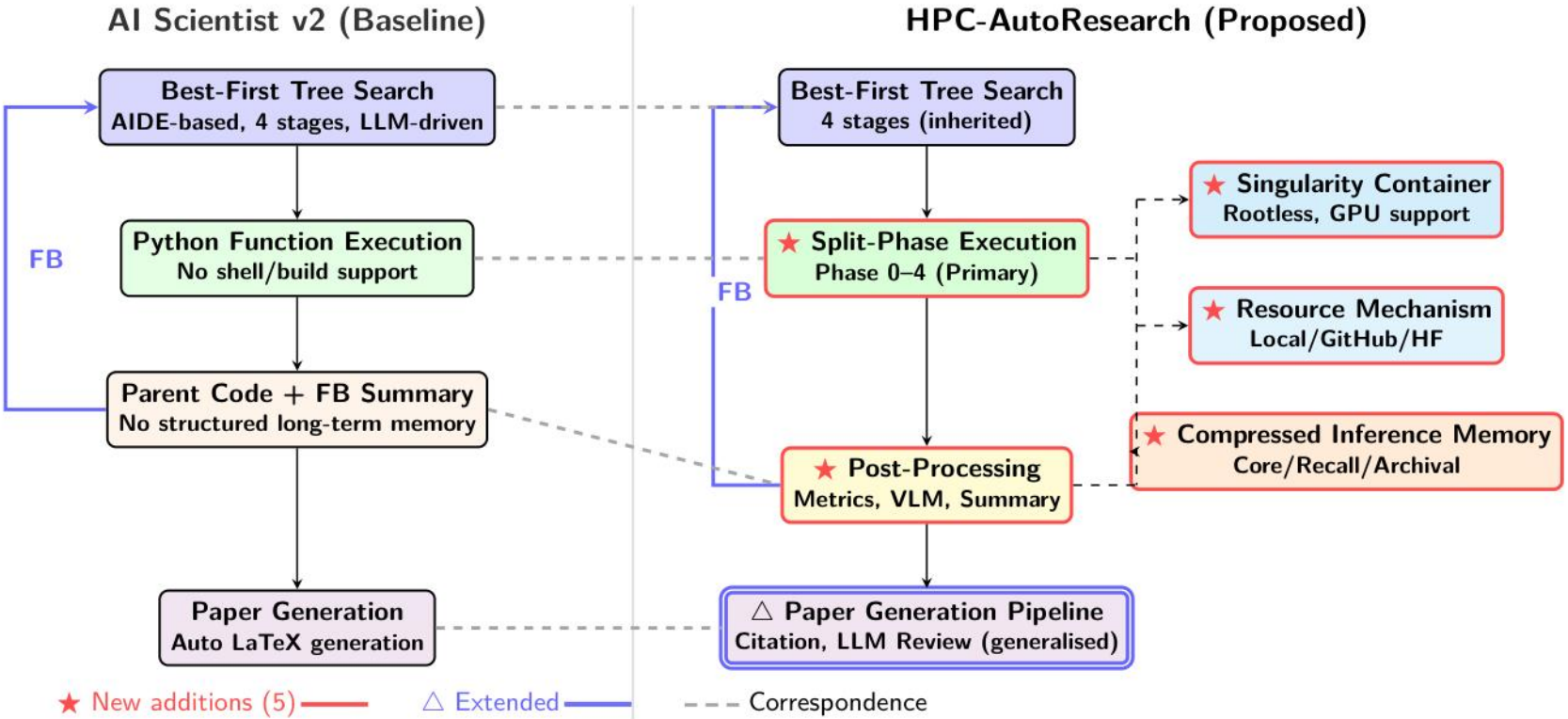
1. **[Primary] Split-Phase Execution Model**
   5 independently repairable phases (planning, env, coding, compile, execute) inside a Best-First tree search — **failure is localised and repaired per phase**.
2. **[Supporting] Compressed Inference Memory**
   3-tier MemGPT extension (Core / Recall / Archival) that **accumulates repair knowledge** across the search, with Copy-on-Write inheritance.
3. **Generalised LLM evaluation**
   ML review prompts rewritten to **domain-agnostic** language — fair evaluation of HPC papers.

System Overview: From a Single Idea to a Reviewed Paper



IDEA document only as input → **fully autonomous** from env setup to paper generation & review.

AI Scientist v2 → HPC-AutoResearch: What We Added

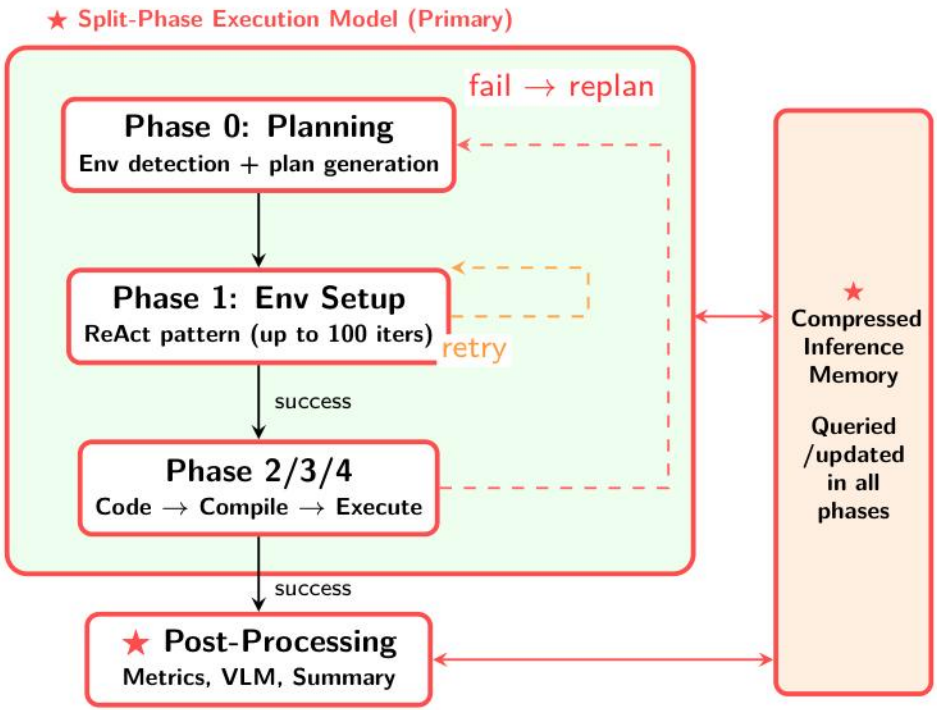


★ Split-Phase Execution Model (Primary Contribution)

- **Phase 0 (Planning)**
  Auto-detect HW/SW env. in the container
- **Phase 1 (Env Setup)**
  ReAct-pattern dep. resolution (up to 100)
  built into per-worker SIF images
- **Phases 2–4**
  Coding → Compilation → Execution
  single LLM call; errors → targeted repair

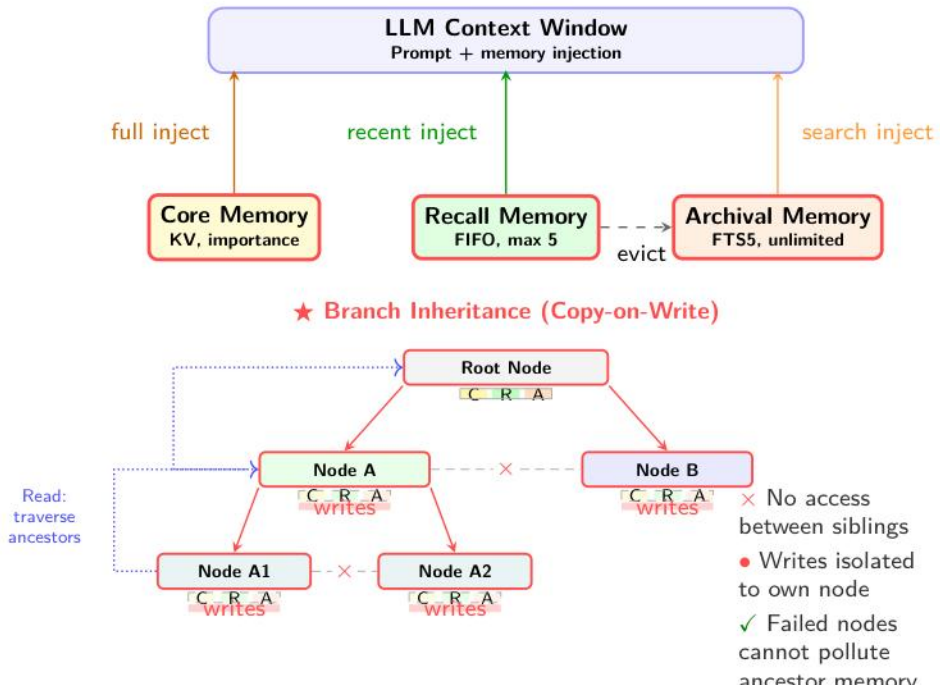
Each phase is **independently repairable** — failures are localised before they propagate.

Tool detail in appendix. ReAct: Yao et al., arXiv:2210.03629, 2023.



★ Compressed Inference Memory (extends MemGPT[2])

- **Core Memory**
  KV store, **always injected**
  importance-based eviction
- **Recall Memory**
  FIFO of recent events (max 5)
  overflow migrates to Archival
- **Archival Memory**
  SQLite FTS5, **unlimited**
  auto-retrieval on any non-zero exit
  (top-k=3)
- **Branch inheritance (Copy-on-Write)**
  children read ancestor memory; **writes isolated** to own branch



[2] Packer et al., "MemGPT: Towards LLMs as Operating Systems," arXiv:2310.08560, 2024.

Experimental Setup

- **Task**: Performance analysis of a **Himeno-style OpenMP stencil proxy**[3] (widely-used CFD Poisson stencil)
- **Cluster (RIKEN)**: dual-socket AMD EPYC 9554 (128 cores), SLURM, Ubuntu 22.04
- **Evaluation**: all outputs run the **same pipeline** (metrics → plots → VLM → paper), scored by a **generalised LLM review** (7 criteria, 0–4)

Condition	Model	Operation & inputs
HPC-AutoResearch (Proposed)	GPT-5.2	**Fully autonomous**, **IDEA only** (∼900 w)
Proposed (w/o memory)	GPT-5.2	Ablation: Compressed Inference Memory off
Claude Code	Claude Opus 4.6	**Manual** CLI; IDEA + TASK (∼500 w)
Codex CLI	GPT-5.2-Codex	**Manual** CLI; IDEA + TASK
Gemini CLI	Gemini 3 Pro Preview	**Manual** CLI; IDEA + TASK

CLI agents also get an **explicit TASK document** → **informational advantage** over ours.

[3] Himeno, "Himeno Benchmark," RIKEN, 2004.

Results: Proposed System Hits 3/4 on All 7 Criteria — CLI Agents Do Not

	Proposed	**Proposed (w/o mem)**	Claude	Codex	Gemini
Originality	**3**/4	**3**/4	2/4	2/4	2/4
Quality	**3**/4	**3**/4	**3**/4	1/4	2/4
Clarity	**3**/4	**3**/4	**3**/4	2/4	2/4
Significance	**3**/4	**3**/4	2/4	2/4	**3**/4
Soundness	**3**/4	**3**/4	**3**/4	1/4	**3**/4
Presentation	**3**/4	**3**/4	**3**/4	2/4	2/4
Contribution	**3**/4	**3**/4	2/4	1/4	**3**/4
Overall	**7**/10	**7**/10	6/10	3/10	**7**/10
Decision	**Accept**	Accept	Accept	**Reject**	Accept
Node success rate	**47.0%**	40.5%	–	–	–
Stage 2 success	**54.2%**	29.2%	–	–	–

- **Codex CLI**: couldn't do SLURM submission → failed at env setup, only **Reject** — native HPC scaffolding is a **prerequisite, not an optimisation**.
- Claude / Gemini: Accept, but lower originality/quality (they had explicit TASK instructions).
- **Memory ablation**: review scores *unchanged*, but node success **47.0** → **40.5**%, Stage 2 **54.2** → **29.2**%.

Full per-criterion scores from the companion paper / Zenodo 18518994; the accepted abstract reports a condensed subset.

The Actual Papers: Ours Fully Autonomous from a 900-Word IDEA

Proposed

Autotuning for Reproducibility: Jointly Optimizing Median Speed and Tail Stability for NUMA Stencil Codes

Anonymous Authors
Anonymous Institution

Abstract

1 Introduction

Overall **7**/10 Accept
All 7 criteria 3/4

Fully autonomous

Claude Code

Performance Stability of the Himeno Benchmark Across Problem Sizes: A Three-Metric Characterization on AMD EPYC

Autocontext Research Pipeline
February 2026

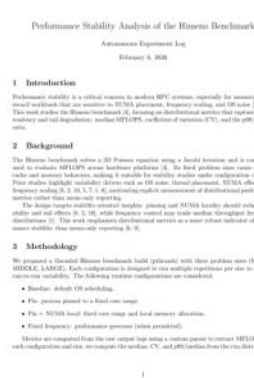
Abstract

1 Introduction

Overall 6/10 Accept
Originality 2/4

Manual operation

Codex CLI



Performance Stability Analysis of the Himeno Benchmark

Autonomous Experiment Log
February 6, 2026

1 Introduction

2 Background

3 Methodology

Overall 3/10 **Reject**
SLURM failure

Manual operation

Gemini CLI

Impact of Thread Affinity on Himeno Benchmark Performance Stability

Gemini Agent
February 6, 2026

Abstract

1 Introduction

2 Methodology

2.1 Experimental Setup

2.2 Metrics

2.3 Configurations

Overall 7/10 Accept
Clarity 2/4

Manual operation

Complete LaTeX papers — abstract, figures, citations, review.
Proposed: **IDEA only, fully autonomous** / others: IDEA + TASK, **manual operation**. Full PDFs: Zenodo 18518994.

Why Memory Matters: Consistent Methodology Across Stages

Per-Stage Metric Availability

St.	Cond.	Med.	CV	p99	Pareto
1	W/ mem	✓	✓	✓	✓
	W/o mem	✓	✓	✓	–
2	W/ mem	✓	✓	✓	✓
	W/o mem	✓	△	✓	–
3	W/ mem	✓	✓	✓	✓
	W/o mem	✓	✓	✓	–
4	W/ mem	✓	✓	✓	✓
	W/o mem	✓	–	–	–

✓: present –: absent △: reinvented methodology

With memory

- **Unified** measurement methodology across all stages
- Cross-stage comparison feasible; metrics improve monotonically

Without memory

- Each stage **reinvented** its methodology — switching between MFLOPS-, CV-, and Pareto-based metrics
- Pareto hypervolume **never computed**; Stage 4 lost CV and p99/med
- Same failure patterns **repeated** across nodes

⇒ Memory contributes to **robustness & knowledge accumulation**, not raw review score.

Discussion

- **Split-Phase makes autonomous HPC research viable**
  Localises failures and accumulates repair knowledge across nodes — exactly the pipeline-wide abort that sank Codex CLI.
- **Targets the infrastructure layer**
  Makes LLM experiments on real clusters possible *at all* — **complementary** to higher-level planning frameworks, not competing.
- **Generalises beyond benchmark kernels**
  With input files + SLURM scripts instead of source: CFD (OpenFOAM, Nek5000), MD (GROMACS, LAMMPS), DFT (Quantum ESPRESSO).
- **Limitations**
  Single benchmark, automated review only, single-run statistics.

Conclusion & Future Work

Conclusion

- **HPC-AutoResearch**: the first autonomous research system adapted for HPC, on top of AI Scientist v2
- **Primary** — Split-Phase execution (5 phases) localises failures and enables iterative repair
- **Supporting** — Compressed Inference Memory: node success 40.5 $\rightarrow$ 47.0%, Stage 2 29.2 $\rightarrow$ 54.2%
- **Only** system achieving 3/4 on all 7 review criteria (fully autonomous, IDEA-only)

Future Work

- Multi-node MPI campaigns
- Human-expert evaluation to complement automated LLM review
- Integration with scientific workflow managers (Snakemake, Nextflow)

Code: `https://github.com/kotama7/HPC-AutoResearch` Data: `https://zenodo.org/records/18518994`

Successor Project: ARI — A More General Autonomous-Research Framework

ARI — Autonomous Research Infrastructure

A **general-purpose, end-to-end** autonomous research pipeline —
generalising the ideas behind HPC-AutoResearch **beyond HPC**.

- Successor project — **code already public**, paper forthcoming

Repository

`github.com/kotama7/ARI`

Homepage

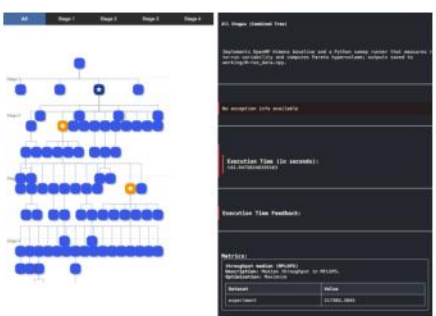
`kotama7.github.io/ARI`

Thank you! I'm happy to take your questions.

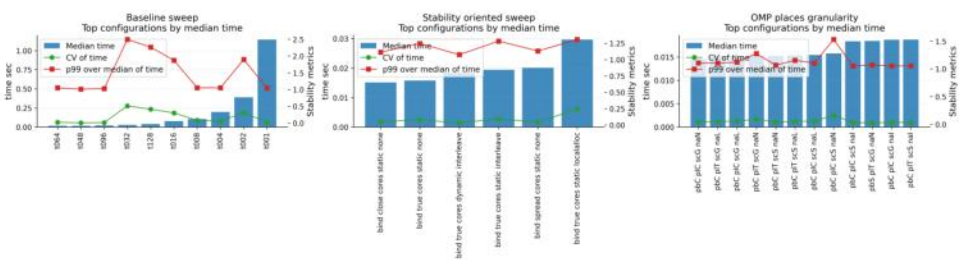
Backup: A Tuning Campaign With No Human in the Loop

From a **single ~900-word hypothesis**, the system **autonomously** designed a multi-dimensional study:

- **3 problem sizes** spanning **L1/L2 cache, LLC, DRAM** (~50 MB / ~400 MB / ~3.2 GB)
- **Pareto-front** over compile-time transforms (loop order, tiling, SIMD, unrolling) & runtime knobs (OpenMP affinity, placement, NUMA)



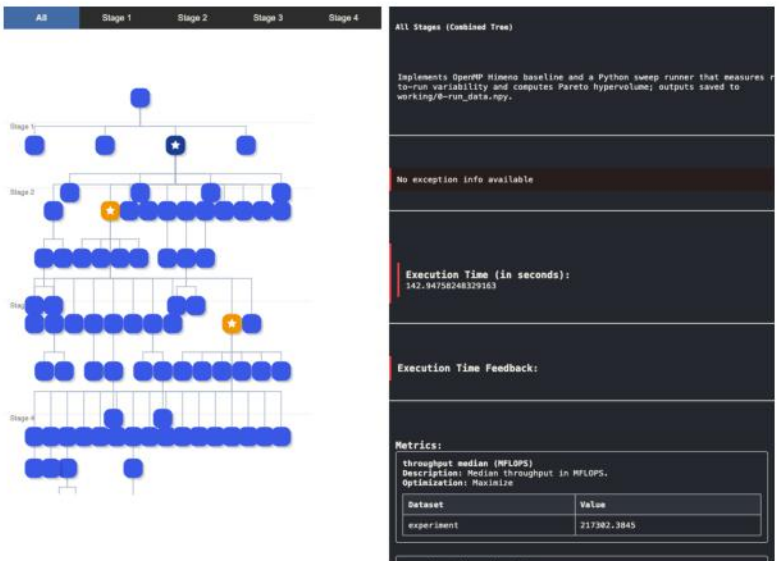
Real Best-First tree run (4 stages)



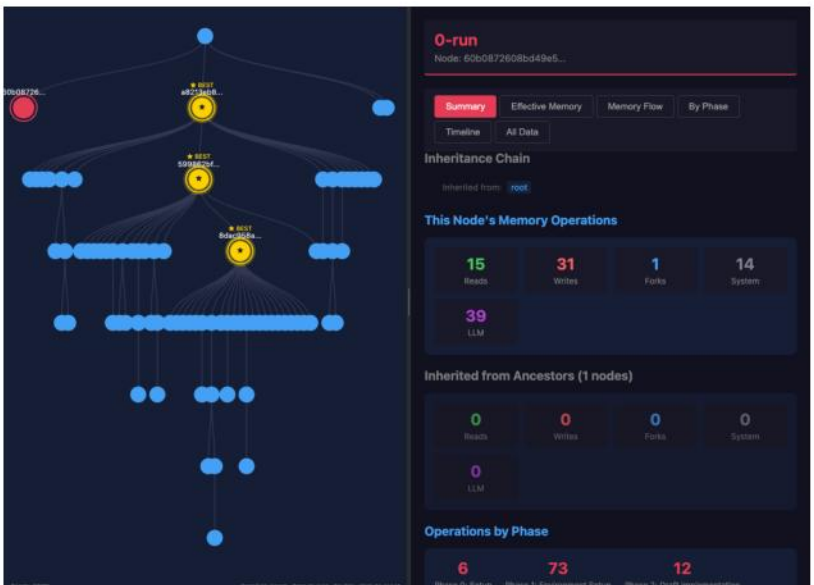
Auto-generated multi-metric stability sweep
(median / CV / p99-over-median; full plots on Zenodo)

Breadth that would need substantial HPC expertise to design manually.

Backup: Best-First Tree Search & Memory Dashboard



Combined tree across Stages 1–4; node panel shows per-node metrics, exceptions, and execution time



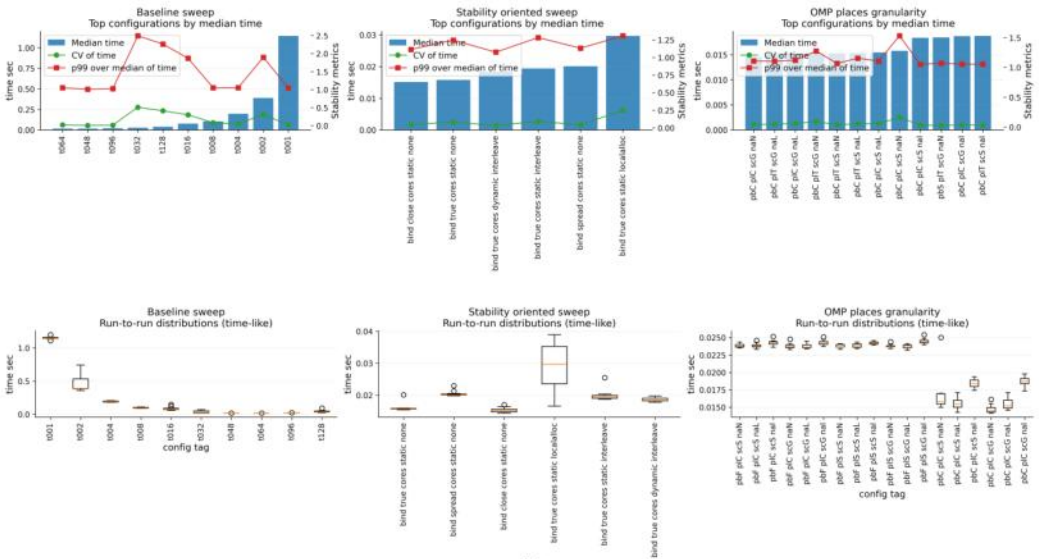
Memory dashboard: per-node Core/Recall/Archival operation statistics across phases

Backup: Phase Boundaries Govern Recovery, Not Call Granularity

- **Phase 0 (Planning)** — per-node HW/SW detection inside the Singularity container: CPU/GPU topology (`lscpu`, `nvidia-smi`), module system (`module avail`), SLURM partitions (`sinfo`, `scontrol`), filesystems (`lfs df`).
- **Phase 1 (Env Setup)** — ReAct-pattern iterative dependency resolution (up to 100 iterations); built dependencies baked into **per-worker SIF images**.
- **Phase 2 (Coding)** — multi-file source as structured JSON (C/C++/CUDA/Fortran, headers, Makefiles).
- **Phase 3 (Compilation)** — `gcc`/`g++`/`nvcc`; captures errors for targeted LLM-driven repair.
- **Phase 4 (Execution)** — submits via `sbatch`, monitors via `squeue`, collects output.

Phases 2–4 are issued in a **single LLM call** for efficiency; the phase *boundaries* govern **failure recovery**, not call granularity. Archival retrieval is triggered by *any* non-zero exit code (top-k=3 "Relevant past errors").

Backup: Autonomously Generated Analysis Plots



Top: per-configuration median / CV / p99-over-median (stability metrics). Bottom: run-to-run time distributions (boxplots).

Generated automatically by the post-processing pipeline across baseline / stability / OpenMP-placement sweeps.