Evaluating LLM Coding Agents on SZ-Family Lossy Compression

Changqing Li, Shouwei Gao, Kai Zhao, Sheng Di, Wenqian Dong

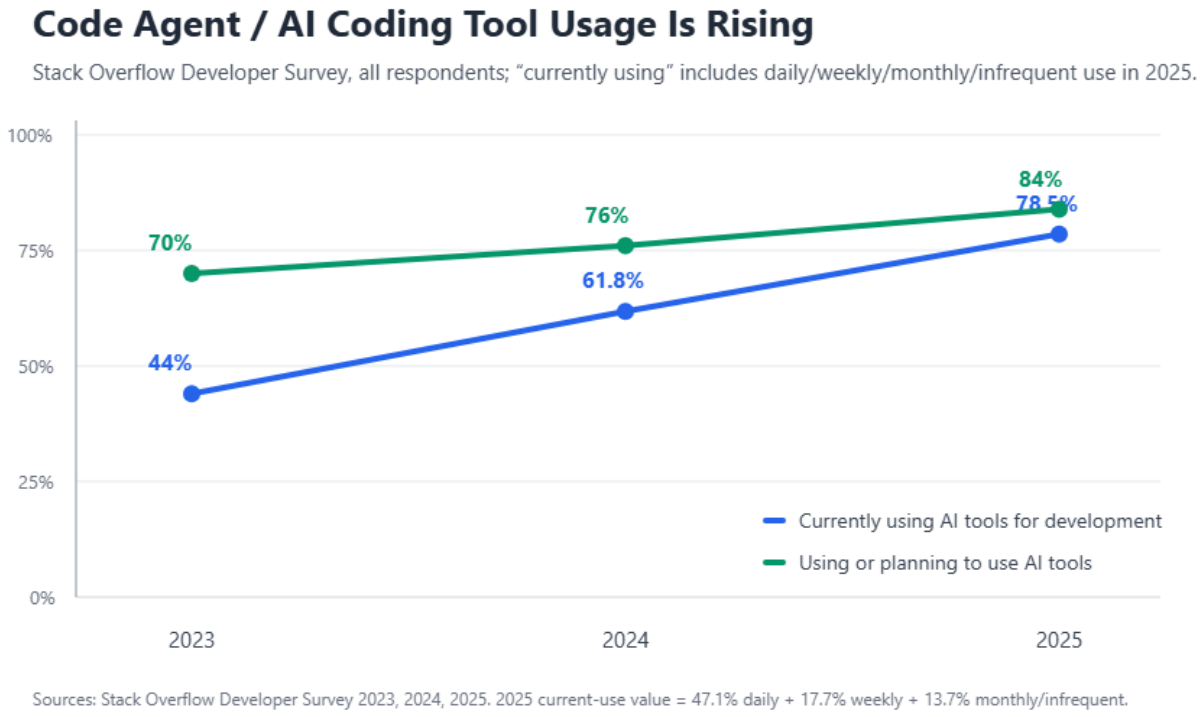
Presenter: Sheng Di

Outline

- Research Background
- Research Motivation (Basic Idea)
- Benchmark Design
- Key Takeaways
- Conclusion and Future Work

Research Background

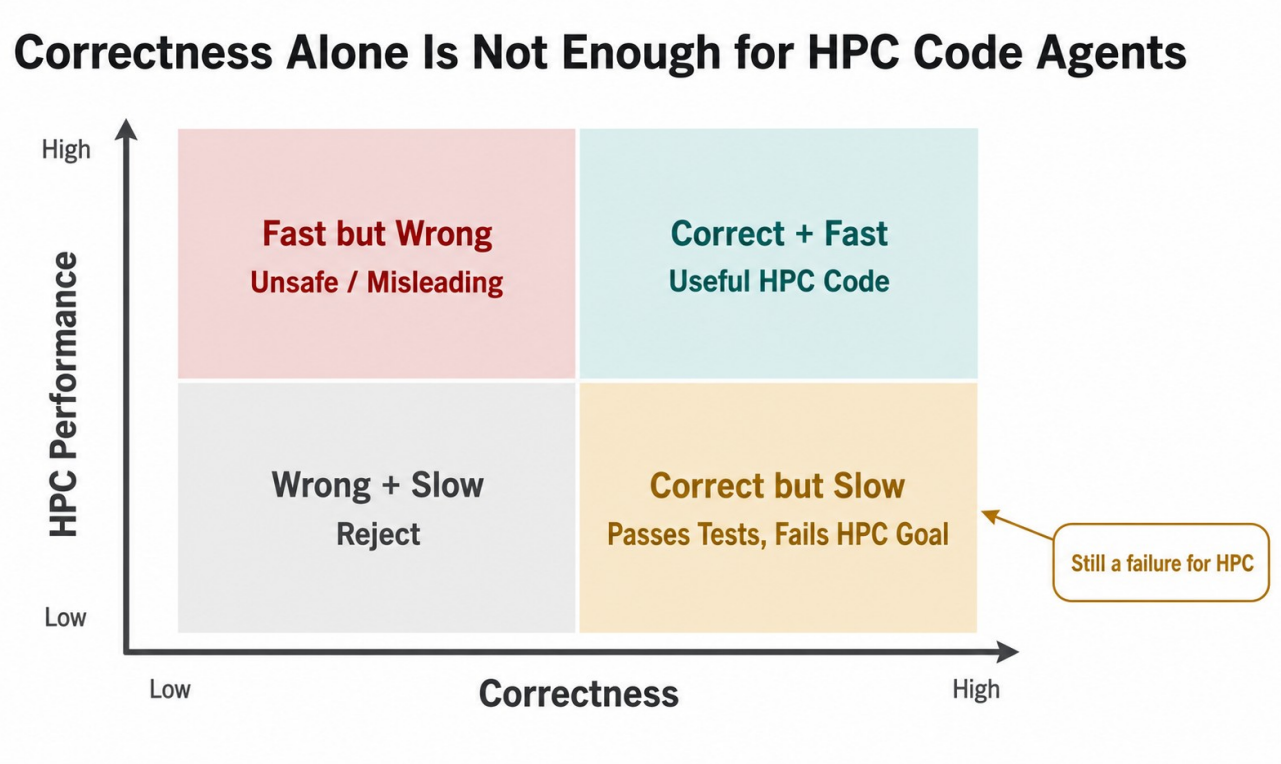
- AI coding tools are rapidly becoming mainstream.
- Adoption grew from 44% (2023) to 78.5% (2025).
- Modern AI agents can generate, run, debug, and refine code iteratively.
- Key HPC question:
  - *Can AI agents generate high-performance scientific code (e.g., CUDA kernels)*?



Data source: Stack Overflow Developer Survey 2023-2025, "AI tools in the development process"; 2025 AI Agents section. Chart recreated by author.

Research Background (Cont'd)

- Why not just code correctness?
  - HPC requires both correctness and performance.
  - Correct but slow is still not enough for HPC.
  - Goal: Correct and fast HPC code generation.
  - Performance is what makes HPC code useful.



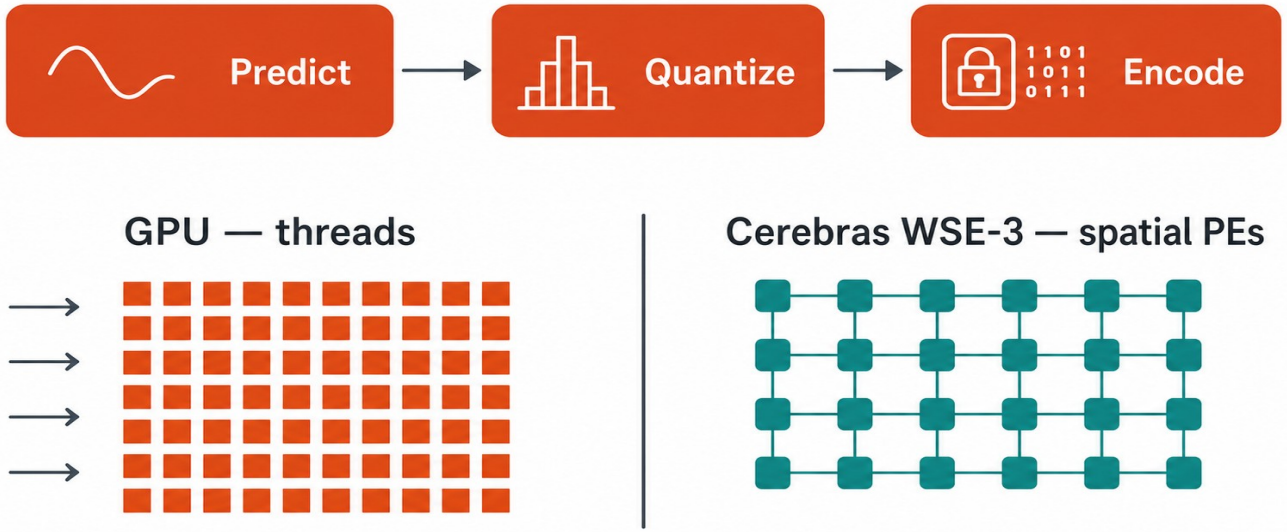
Research Motivation

- The research objective of this work is to evaluate how effectively large language model (LLM) **coding agents** can translate and optimize high-performance computing (HPC) lossy compression codes **across fundamentally different hardware architectures**, including NVIDIA GPUs and Cerebras wafer-scale accelerators.
- The study focuses on **SZ-family error-bounded lossy compression** kernels (SZp and SZx) and investigates not only correctness, but also performance, optimization behavior, convergence patterns, and architecture-specific failure modes of AI-generated code.

Research Motivation (Cont'd)

Why Lossy Compression as case study?

- Real HPC workload for reducing large simulation outputs.
- Clear SZ pipeline: **predict → quantize → encode**.
- Existing CUDA versions of **SZp** and **SZx** give concrete baselines.
- Same algorithm behaves very differently on GPU vs. WSE, making it a strong test of agent optimization.



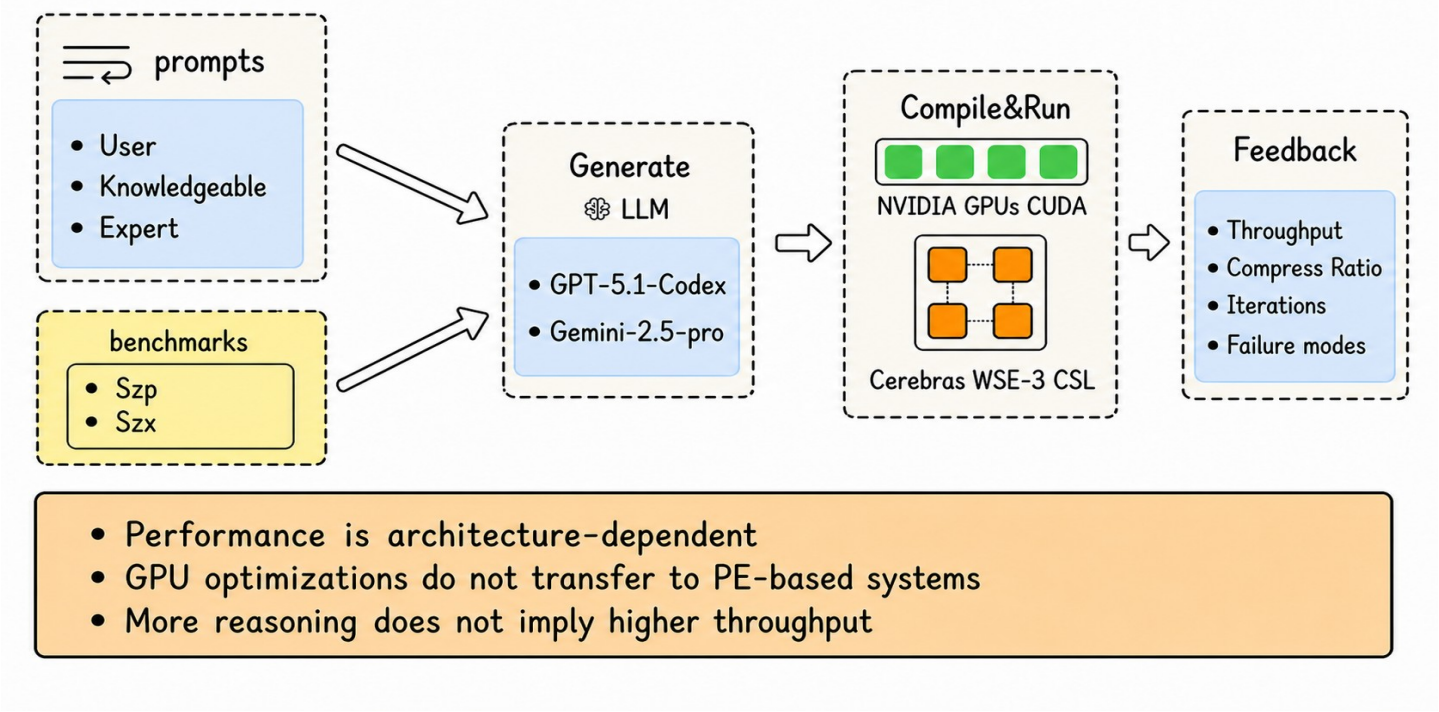
SZp/SZx projects production-level design, with many papers published in top-rank venues: cuSZp (**SC23**), cuSZp2 (**SC24**), cuSZp3 (**SC25**), Szp-CPU (**SC24**), SZx (**HPDC22**), CereSZ (**HPDC24**), CereSZ2 (**IPDPS25**), CereSZ3/P3Z (**IPDPS26**)

Challenge: GPU and Cerebras demand fundamentally different optimization strategies for the same kernel.

Benchmark Design

Two frontier coding agents:
GPT-5.1-Codex and Gemini-2.5-pro
Three prompt levels:
User, Knowledgeable, Expert
Two HPC platforms:
NVIDIA V100 (CUDA) and Cerebras CS-3 (CSL)
Agentic loop:
generate → compile → run → feedback
Metrics:
performance, convergence, and failure modes

User: Basic request with minimal guidance (e.g., “make this code faster”).
Knowledgeable: Adds general domain and optimization context, such as SZ pipeline stages and performance concepts.
Expert: Provides architecture-specific optimization guidance, such as GPU memory coalescing or Cerebras spatial mapping and communication hints.



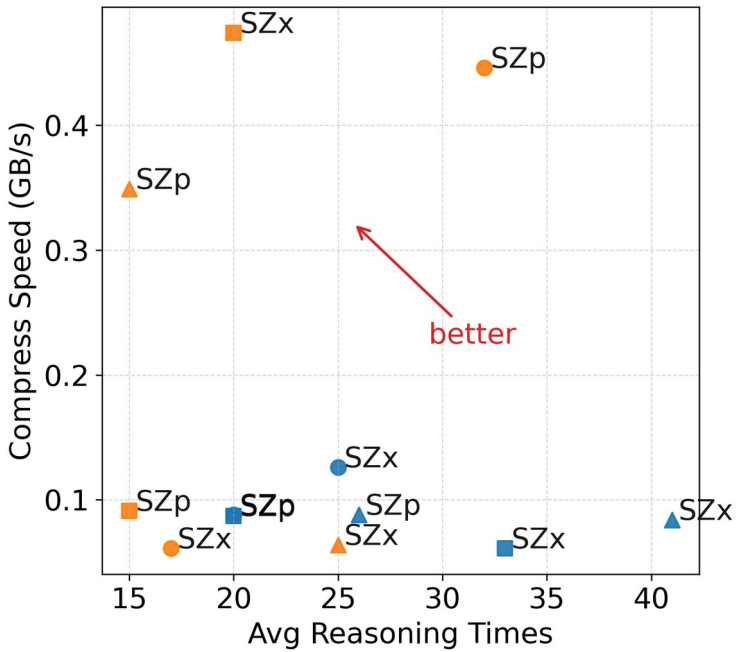
Key Takeaways

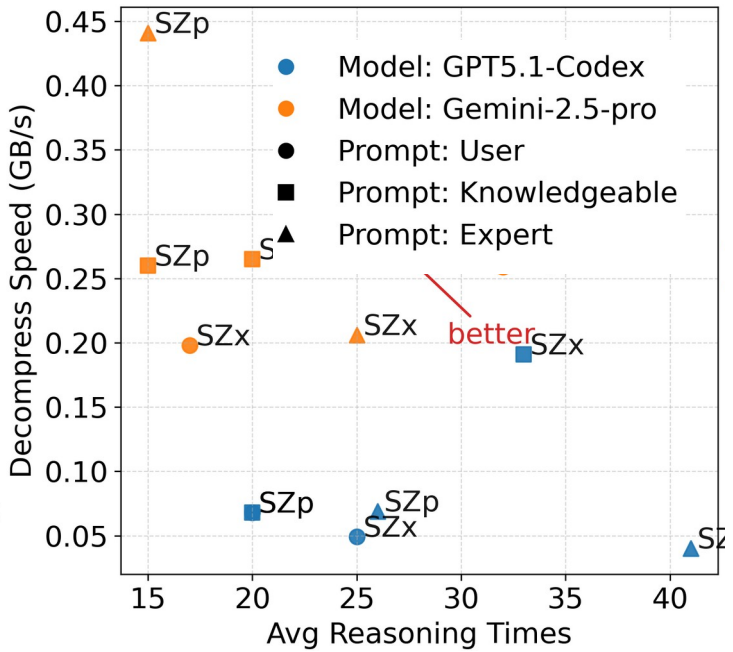
- **Correctness is just the start**
- **More reasoning ≠ better code**
- **Prompts matter for some models**
- **GPU wins do not transfer**

Implication: evaluating LLM coding agents for HPC must be cross-architecture.

Key Takeaways (Cont'd)

- ***GPU: More Reasoning Does Not Necessarily Mean Faster Code***
  - *Gemini-2.5-pro* clusters around 15-25 iterations and reaches higher throughput.
  - *GPT-5.1-Codex* often uses 30-41 iterations but produces slower code.
  - Extra *GPT* iterations mostly go into compiler fixes and correctness debugging, not optimization.



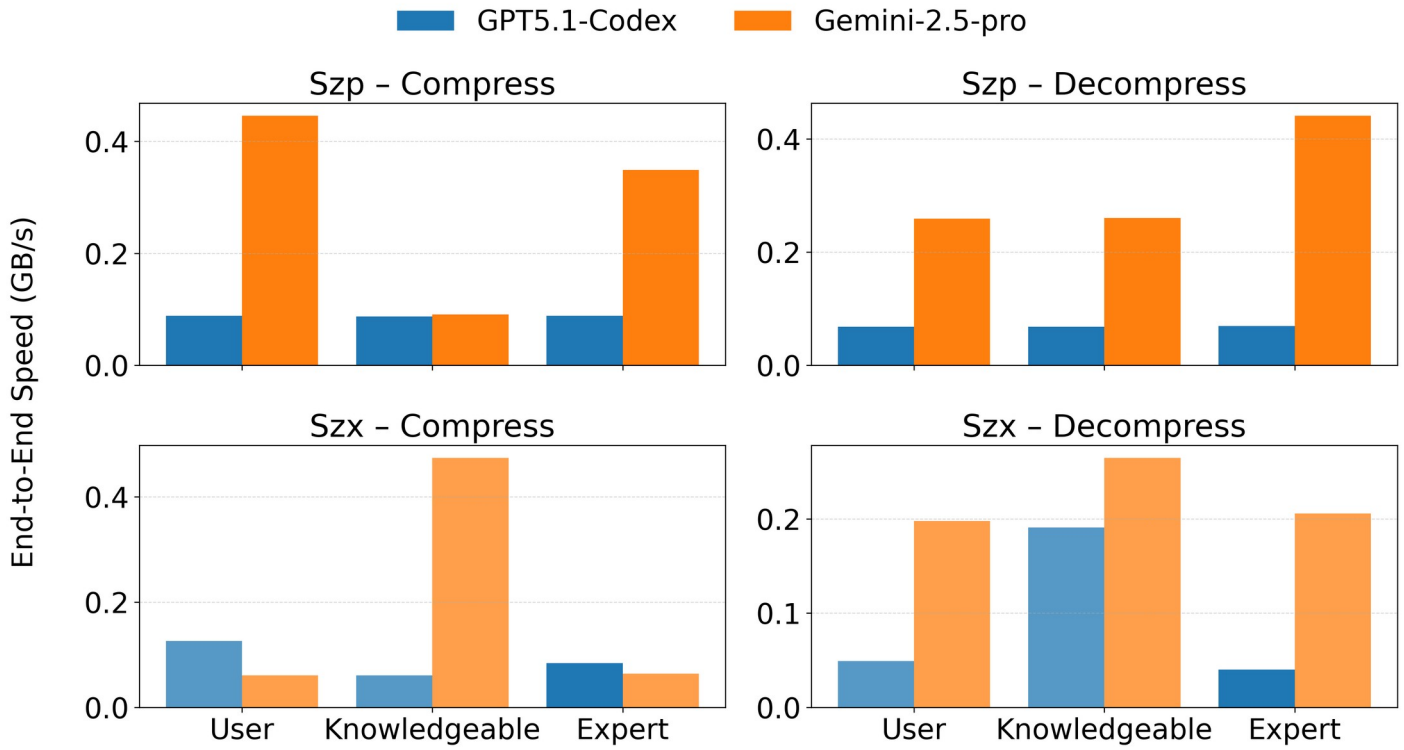


Reasoning iterations are not a proxy for code quality

They often indicate how much trouble the model had before producing final code

Key Takeaways (Cont'd)

- ***GPU: Model and Prompt Matter***
  - *Gemini-2.5-pro* reaches higher peak throughput on V100.
  - *Gemini-2.5-pro* is highly prompt-sensitive: better prompts lead to much faster code.
  - GPT-5.1-Codex is more stable across prompt tiers, but has a lower ceiling.
  - This creates a tradeoff: high ceiling + high variance vs. low ceiling + low variance.



Prompt sensitivity is model-dependent, not just task-dependent.

Key Takeaways (Cont'd)

Cerebras: GPU Winners Fail on Spatial Accelerators

GPU-successful models can fail on Cerebras on **Cerebras**, runnability matters more than throughput.

- "134 s" = runnable execution;
- "0 s" = execution failure

Many LLM-generated programs failed to produce runnable executions at all due to:

- incorrect host-to-PE communication,
- invalid spatial mappings,
- synchronization problems,
- static dataflow violations.

Model	Task	Prompt	Reasoning Iter.	Exec. Time (s)
GPT-5.1-Codex	SZp	User	74	134
GPT-5.1-Codex	SZp	Knowledgeable	100 †	134
GPT-5.1-Codex	SZp	Expert	79	134
GPT-5.1-Codex	SZx	User	100 †	134
GPT-5.1-Codex	SZx	Knowledgeable	99	134
GPT-5.1-Codex	SZx	Expert	100 †	134
Gemini-2.5-pro	SZp	User	100 †	0 ‡
Gemini-2.5-pro	SZp	Knowledgeable	37	0 ‡
Gemini-2.5-pro	SZp	Expert	100 †	0 ‡
Gemini-2.5-pro	SZx	User	9	0 ‡
Gemini-2.5-pro	SZx	Knowledgeable	11	0 ‡
Gemini-2.5-pro	SZx	Expert	12	0 ‡

† reasoning budget exhausted. ‡ failed to generate runnable execution.

Conclusion and Future Work

- LLM coding agents can optimize HPC compression kernels, but performance is highly **architecture-dependent**.
- **GPU**-oriented optimization strategies do not transfer well to **Cerebras** spatial accelerators.
- In iterative compile-and-feedback workflows, **additional reasoning iterations** are often spent on debugging and correctness rather than discovering bandwidth-critical optimizations.
- Future work will explore more HPC workloads, additional hardware platforms, and hybrid/profile-guided agent workflows for robust architecture-aware optimization

Acknowledgments: This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under contract DE-AC02-06CH11357, and by the U.S. National Science Foundation under grants 2514351, 2505118, 2514036, and 2513768.