Towards Expert-Level AI Reasoning for Computer Systems and Architecture

AI systems have achieved success in many domains, improving productivity. Tools like Cursor have brought significant gains to programming in mainstream languages. The computer systems community anticipates a similar ChatGPT moment: an AI that can significantly reduce system engineers' workload and improve automation, such as automatically optimizing performance for given workloads or designing hardware accelerators based on algorithm requirements. Such capabilities would save countless engineering hours and transform chip development. However, we have not witnessed this moment yet. LLMs still struggle with low-resource languages like Verilog [1], and cannot optimize code efficiency like human engineers [2].

This gap is not an engineering problem. It is a scientific research problem that requires a paradigm shift. The current paradigm of pretraining large models on internet-crawled data has proven inadequate for this domain. I want to point out three reasons. First, from a domain-specific perspective, computer systems require specialized knowledge and reasoning workflows. For example, an experienced engineer knows to profile the system, identify bottlenecks, then apply targeted optimizations. Neither this knowledge nor these patterns are well-represented on the internet, so LLMs cannot generalize to these problems. Second, from an algorithmic perspective, LLMs can only interpolate within their training distribution. Human engineers solve novel problems by composing skills in new combinations, but current models cannot do this. Third, from an efficiency and cost perspective, large model intelligence is too expensive to deploy at scale, with prohibitive costs that prevent widespread adoption.

My research vision is to advance AI reasoning for computer systems and architecture. To pursue this, I have prepared from three perspectives. Due to space constraints, I highlight key contributions below; my full publication list is on my website.

To address the domain-specific challenge, I am developing machine learning methods specifically for computer systems and architecture. I have contributed to multiple efforts: (1) Teaching Fellow for CS249r and co-authoring a survey, (2) co-developing QUARCH, a community benchmark of 2,600 exam questions from undergraduate and graduate courses across multiple universities, addressing the lack of domain-specific datasets for evaluating LLM reasoning in computer architecture, (3) mentoring an undergraduate on ML for architecture mapping. Most importantly, (4) I am building open-source domain-specific foundation models for computer architecture. Currently, NVIDIA's ChipNeMo [3] is the only domain-adapted LLM in this field, but it remains closed-source. To address this gap, we curated 1.3 billion tokens from four decades of architecture research papers and technology documents. We are adapting foundation models through continued pretraining: an embedding model for domain-specific retrieval, which can serve as knowledge memory for architecture-focused AI agents, and a GPT-based model for domain reasoning. Early results show improved retrieval quality compared to general-purpose embeddings. Leading the training effort, I have gained hands-on experience with data curation, training infrastructure, and hyperparameter configuration.

To address the algorithmic challenge, I am exploring new paradigms for improving LLM reasoning beyond simple scaling. Rather than training larger models or collecting more data, I am investigating how to extract more capability from existing models at inference time. I led the development of SLM-MUX, a training-free framework for combining multiple small language models (SLMs). We initially tried approaches that work on larger models, such as having models debate or discuss with each other. However, these methods fail on SLMs because they have not been exposed to

data representing complex cognitive behaviors during pretraining; they simply lack this capability. At the same time, we observed that different SLMs, trained on their own corpora, tend to excel at complementary sets of problems. This raised a natural question: how can we leverage this complementarity? We found a simple but effective solution: query each model independently and select the output with the highest self-consistency, measured via majority voting or embedding similarity. Combining a 7B and 24B model this way outperforms 72B models from the same period on MATH, GPQA, GSM8K, and HumanEval. This suggests that test-time composition can be a promising alternative to expensive pretraining. I am also contributing to related work on compositional AI profiling, which examines the system implications of such approaches.

To address the efficiency and cost challenge, I bring experience in making AI systems practical to deploy. I have worked on model compression and its co-optimization with hardware design. My DATE 2022 and TCAD 2023 papers combined hardware architecture design space exploration (DSE) with neural architecture search (NAS), and my DAC 2024 paper integrated quantization with hardware DSE. I have also contributed preprints on compression methods, including one combining NAS with AI safety and another evaluating LLM quantization in long-context settings. This background positions me to explore how efficient AI can enable practical deployment of reasoning capabilities at scale.

In summary, my unique strength lies in having deep experience in both systems and machine learning. I have deeper ML understanding than most systems researchers, and more familiarity with systems problems than most AI researchers.

Beyond computer systems, I believe my combination of systems and AI expertise can contribute to other scientific and engineering domains as well.

References

[1] Pinckney et al. "Comprehensive Verilog Design Problems." *arXiv:2506.14074* (2025).

[2] Ma et al. "SWE-fficiency: Can LLMs Optimize Real-World Repositories?" *arXiv:2511.06090* (2025).

[3] Liu et al. "ChipNeMo: Domain-Adapted LLMs for Chip Design." *arXiv:2311.00176* (2024).