

REASONING COMPILER: LLM-Guided Optimizations for Efficient Model Serving

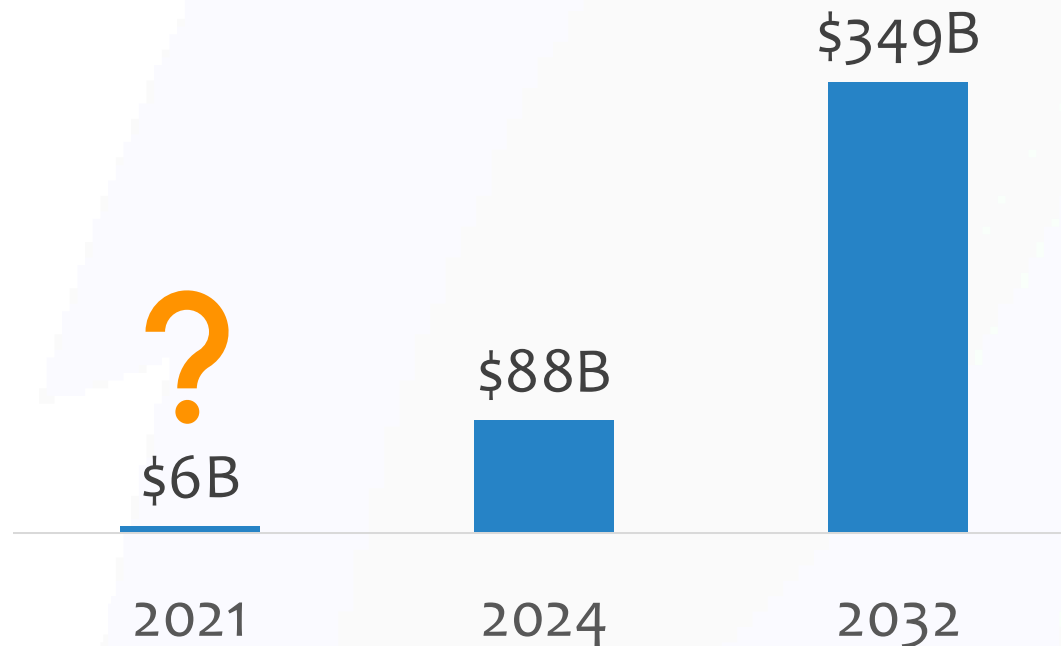
Sujun Tang, Christopher Priebe, Rohan Mahapatra, Lianhui Qin,
Hadi Esmaeilzadeh

Alternative Computing Technologies (**ACT**) Lab
University of California, San Diego



The Shift towards Model Serving

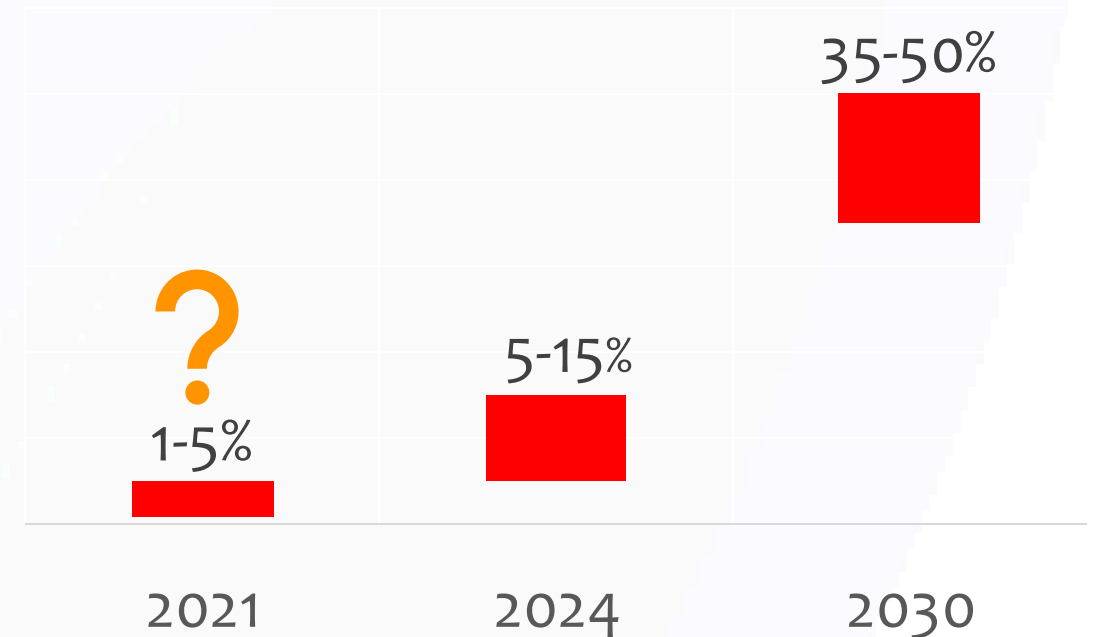
Global Inference Market



Global AI Inference Market is projected to grow at a CAGR of 18.91% over 2025-2032.

Source: Yahoo Finance

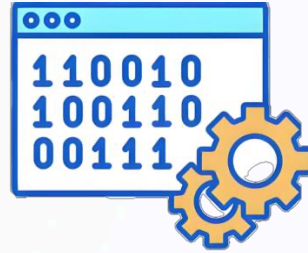
AI Data-Center Power Usage



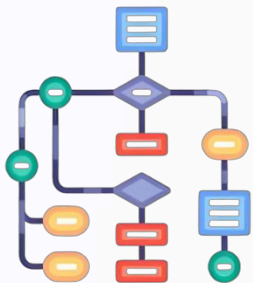
AI-oriented IaaS will more than double in 2025-2026; inference is the demand driver.

Source: Gartner

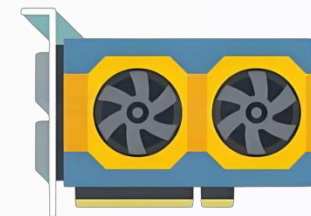
The Triangle of Improvement



Compiler

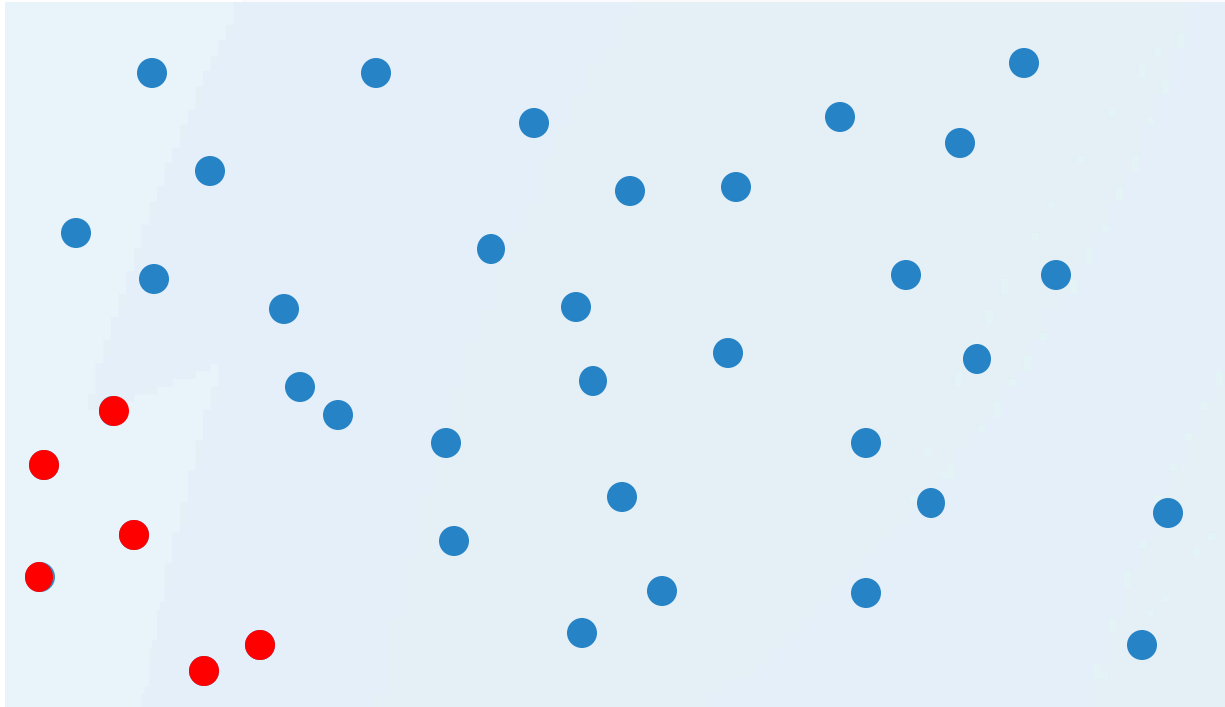


Algorithm



Hardware

Category I: Rule-Based Compiler Optimizations



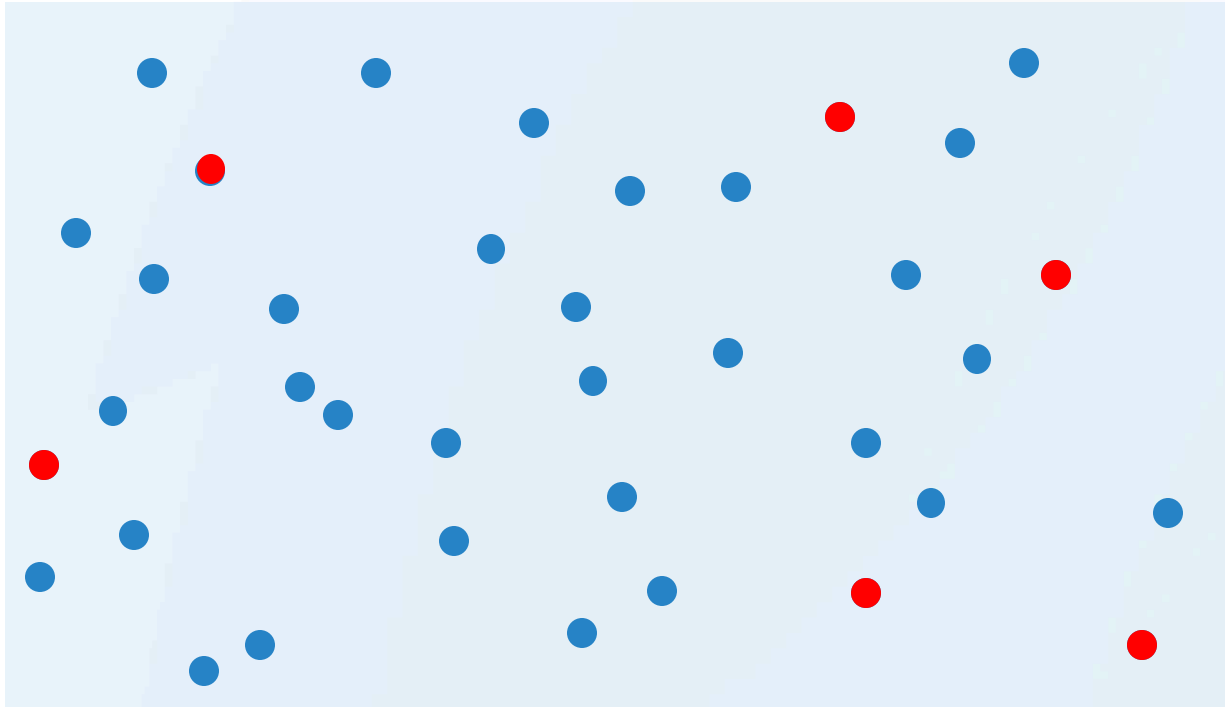
Relies on hand-tuning or domain-specific heuristics

Often overfit to a specific workload or hardware target

+ Relatively Fast

- Cannot explore the entirety of search spaces

Category II: Stochastic Search for Compilation



- + Find Higher Quality Optimized Programs
- Sample Inefficient
- Cannot leverage context and interdependence

STOKE (Stochastic Super Optimization) [ASPLOS '13]

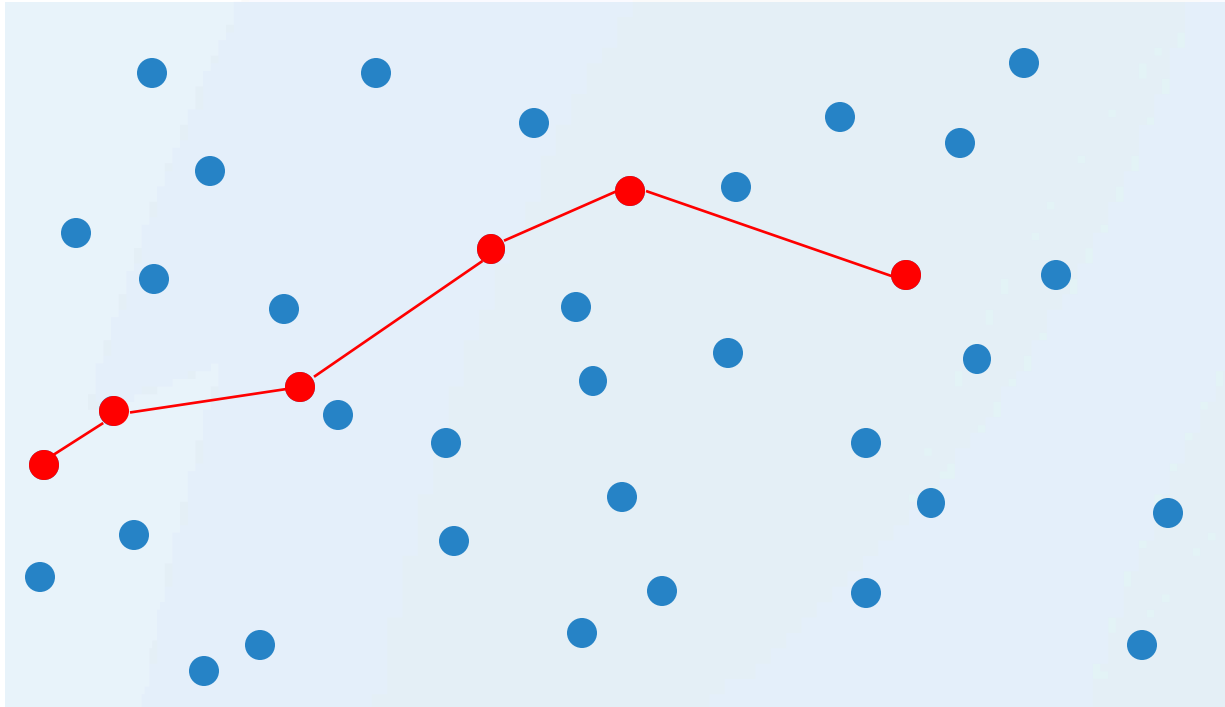
- Markov Chain Monte Carlo (MCMC)
- High quality programs often lie in regions separated by low-probability paths

TVM [OSDI '18], Anson [OSDI '20], FlexTensor [ASPLOS '20], Tensor Comprehensions [arXiv '18]

- Genetic Algorithm
- Simulated Annealing

REASONING COMPILER

Context-Informed, Guided, Structured Search



+ Find Higher Quality Optimized Programs

+ Sample Efficient

Leaps from fully stochastic to structured optimizations

Models optimization as **Markov Decision Process**

Uses **Monte Carlo Tree Search (MCTS)** as a planner

Utilizes **LLM reasoning** as a guide for Monte-Carlo Tree Search

**Can LLM reasoning, without retraining,
guide context-sensitive compiler
optimizations?**

Problem Statement – Neural Code Optimization

```
Self-Attention Layer of Llama-3-8B
...
...
for b in range(1):
    for i in range(16):
        for j in range(4096):
            C[b][i][j] = 0
            for k in range(4096):
                C[b][i][j] += A[b][i][k] * B[k][j]
...
...
```

Tile

Tile size for each loop

Parallelize

Axis, Policy

Vectorize

Width

Unroll

Factor

PackB

k-panel size, j-panel Size

Prefetch

Target, Distance

RegisterBlock

m-register Tile Size,
n-register Tile Size

...

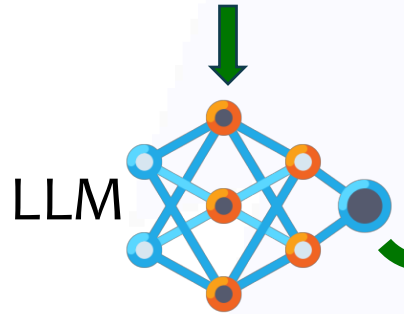
Prompt: Select a Transformation ...

- Code, **Cost** and Transformation History of Current Program
- Code, Cost and Transformation History of Parent Program
- Set of All Possible Transformations

```

...
for b in range(1):
  for i in range(16):
    for j in range(4096):
      c[b][i][j] = 0
    for k in range(4096):
      c[b][i][j] += A[b][i][k] * B[k][j]
...

```



Action₄ ← Tile(.)

Action₄: Tile(.)

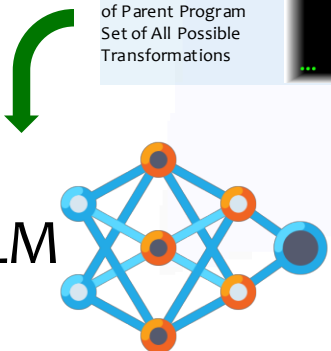
Prompt: Select a Transformation ...

- Code, **Cost** and Transformation History of Current Program
- Code, Cost and Transformation History of Parent Program
- Set of All Possible Transformations

```

...
for b in range(1):
  for i in range(16):
    for j in range(4096):
      c[b][i][j] = 0
    for k in range(4096):
      c[b][i][j] += A[b][i][k] * B[k][j]
...

```



Action₆ ← Unroll(.)

Action₆: Unroll(.)

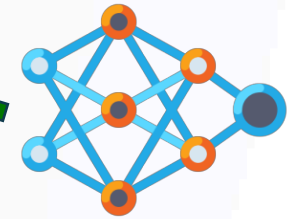
Prompt: Select a Transformation...

- Code, **Cost** and Transformation History of Current Program
- Code, Cost and Transformation History of Parent Program
- Set of All Possible Transformations

```

...
for b in range(1):
  for i in range(16):
    for j in range(4096):
      c[b][i][j] = 0
    for k in range(4096):
      c[b][i][j] += A[b][i][k] * B[k][j]
...

```



Action₅ ← Vectorize(.)

Action₅: Vectorize(.)

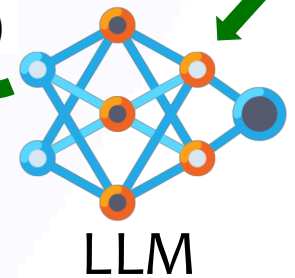
Prompt: Select a Transformation...

- Code, **Cost** and Transformation History of Current Program
- Code, Cost and Transformation History of Parent Program
- Set of All Possible Transformations

```

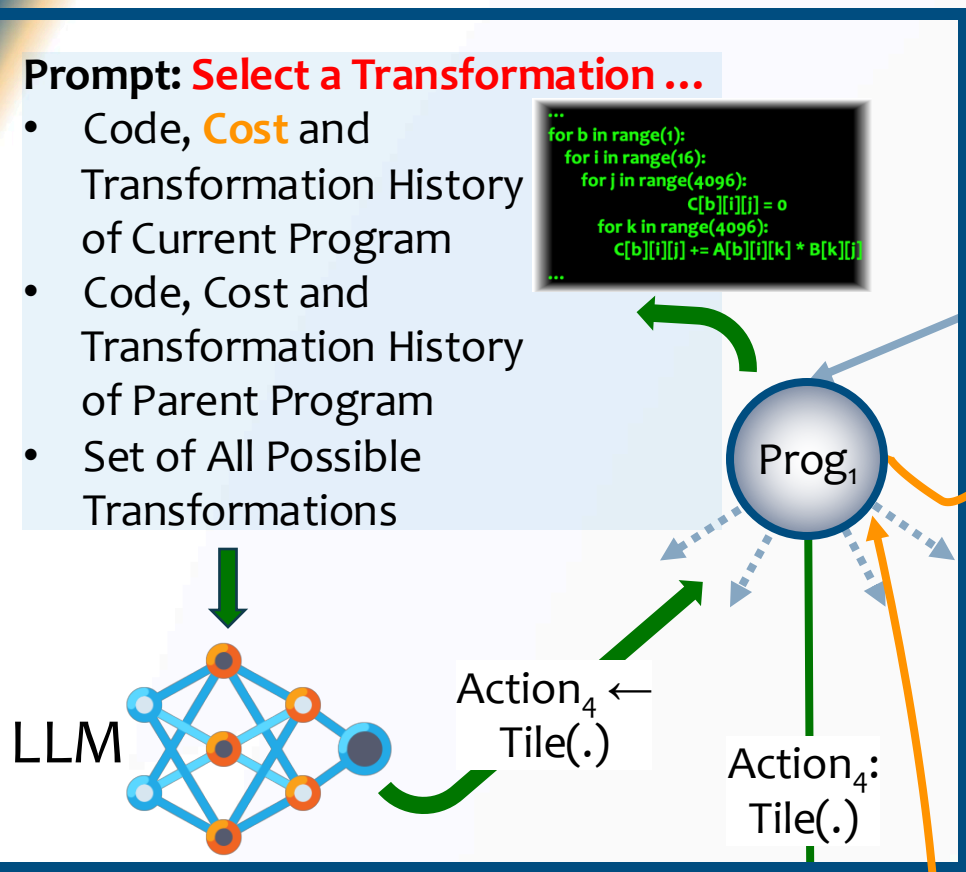
...
for b in range(1):
  for i in range(16):
    for j in range(4096):
      c[b][i][j] = 0
    for k in range(4096):
      c[b][i][j] += A[b][i][k] * B[k][j]
...

```



Action₇ ← PackB(.)

Action₇: Tile(.)



Action₁: Tile(.)

Prog₀

Prog₂

Action₂: Parallelize(.)

Action₃: RegisterBlock(.)

Prog₃

Prog₅

Prog₄

Prog₆

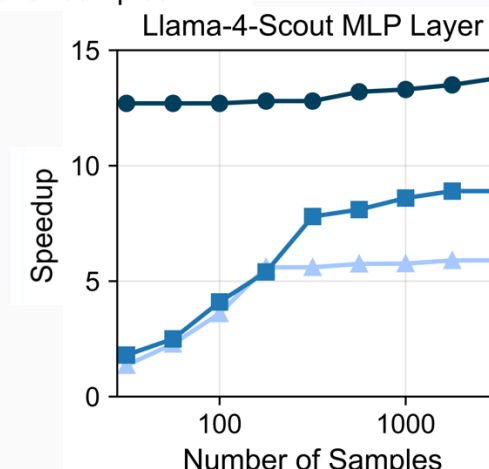
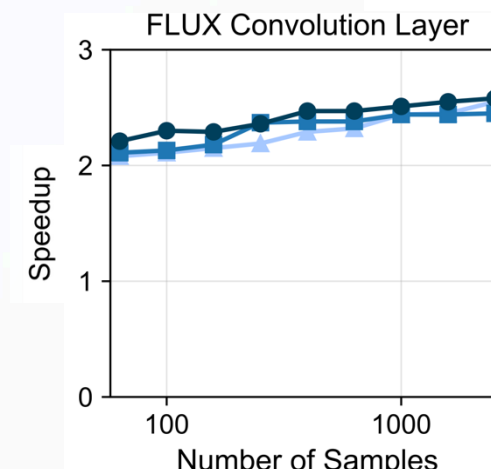
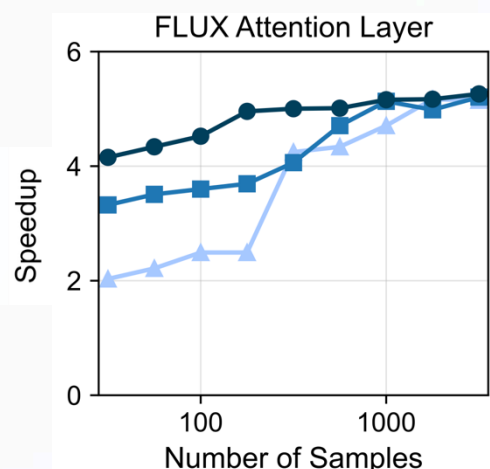
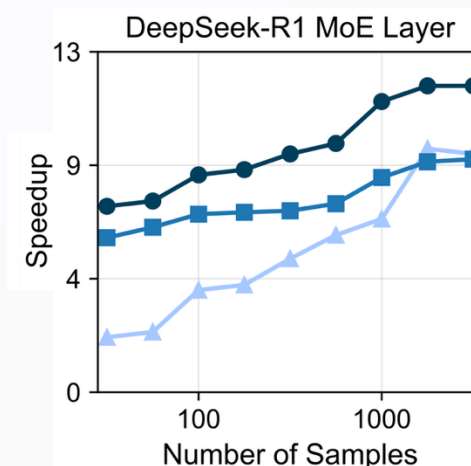
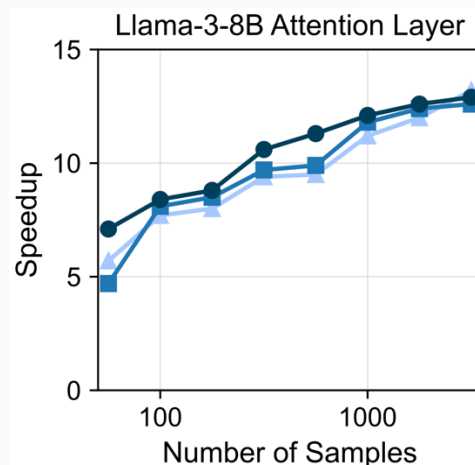
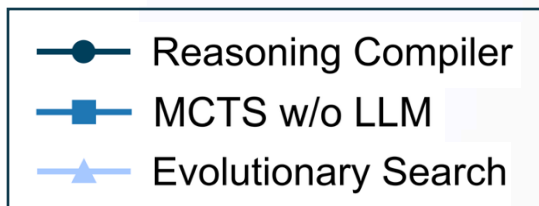
Prog₇

Benchmarks

- Self-Attention Layer from Llama-3-8B
- Mixture-of-Experts Layer from DeepSeek-R1
- Self-Attention Layer from FLUX (Stable Diffusion)
- Convolution Layer from FLUX (Stable Diffusion)
- MLP Layer from Llama-4-Scout
- End-to-End Llama-3-8B

- Amazon Graviton2
- AMD EPYC 7R13
- Apple M2 Pro
- Intel Core i9
- Intel Xeon E3

Higher Speed with Fewer Samples



Across all **25** platform-operator pairs, **REASONING COMPILER** achieves **5.0 ×** speedup with **5.8 ×** fewer samples: **10.8 ×** improvement in sample efficiency over Evolutionary Search.

Consistent End-To-End Improvements

Hardware Platforms	Evolutionary Search		Reasoning Compiler		Improvement	
	# Samples	Speedup	# Samples	Speedup	Sample Reduction	Sample Efficiency Gain
Amazon Graviton2	4,560	3.7×	1,440	5.1×	3.2×	4.4×
AMD EPYC 7R13	410	2.0×	140	2.2×	2.9×	3.2×
Apple M2 Pro	4,820	2.2×	1,770	3.9×	2.7×	4.8×
Intel Core i9	3,800	2.2×	720	4.9×	5.3×	11.8×
Intel Xeon E3	4,640	5.0×	670	5.0×	6.9×	6.9×
Geomean	-	2.8×	-	4.0×	3.9×	5.6×

For Llama-3-8B, **REASONING COMPILER** achieves **4.0×** speedup using **3.9×** fewer samples achieving **5.6×** sample efficiency over Evolutionary Search

REASONING COMPILER

Structured, Sample-Efficient Search

REASONING COMPILER represents an effective **leap** from stochastic search to **LLM-guided, structured** planning for compiler optimizations.

REASONING COMPILER formulates optimizations as a **sequential, context-aware** planning process, pairing LLM-generated proposals with **MCTS**.

REASONING COMPILER's sample efficiency lowers serving cost, reduces energy, improves system responsiveness, and accelerates training cycles.

The same LLM that guides compilation can accelerate its own inference, creating a **virtuous, self-optimizing cycle**.

Paper

<https://arxiv.org/abs/2506.01374>



Github

https://github.com/Anna-Bele/LLM_MCTS_Search



REASONING COMPILER: LLM-Guided Optimizations for Efficient Model Serving

Alternative Computing Technologies (**ACT**) Lab
Contact: Sujun Tang, sujun@ucsd.edu