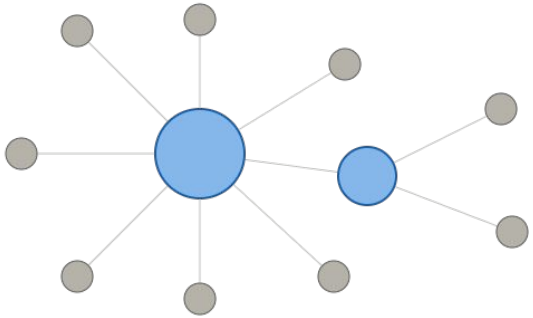
ParEVO: Synthesizing Code for Irregular Data: High-Performance Parallelism through Agentic Evolution

Liu Yang*[1], Zeyu Nie*[1], Andrew Liu[1], Felix Zou[1], Deniz Altinbuken[2], Amir Yazdanbakhsh[2], Quanquan C. Liu[1]

[1] Department of Computer Science, Yale University, New Haven, CT, USA
[2] Google DeepMind, Mountain View, CA, USA

The Problem: Irregular Workload

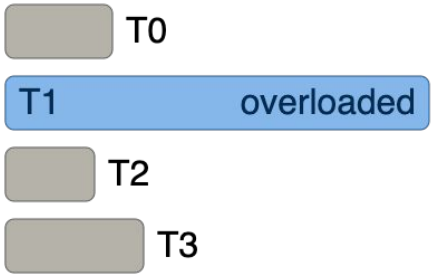
Irregular graph
Work per vertex is data-dependent



High-degree hubs (blue) =
most of the work

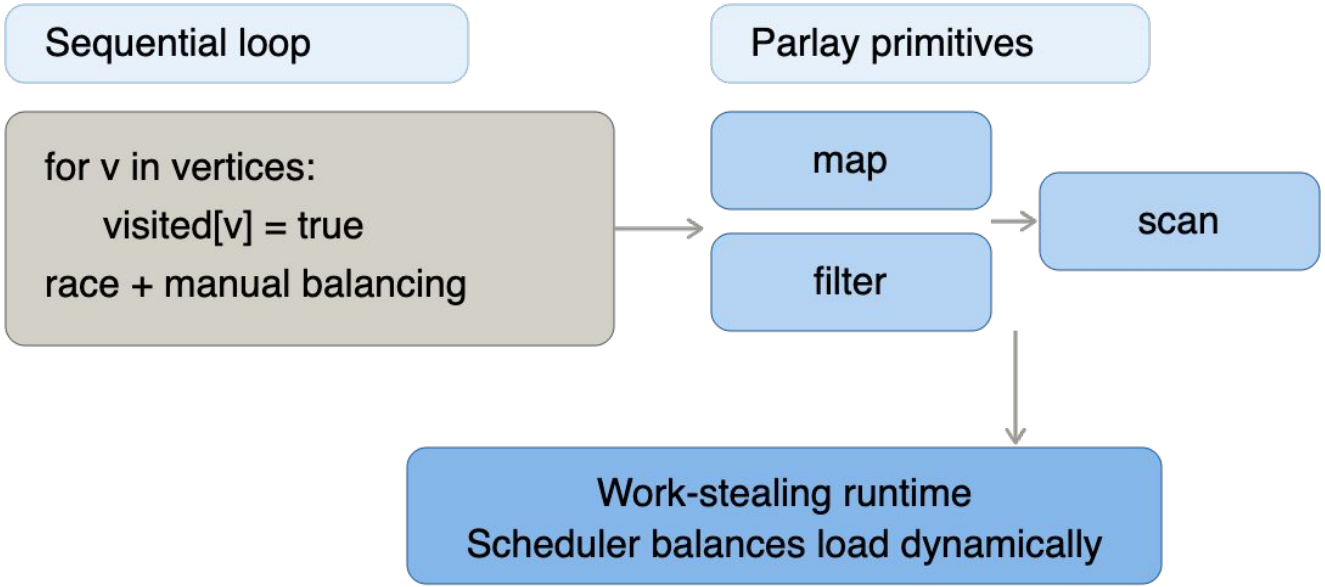
static split

Threads (fixed partition)
Even iterations, uneven work



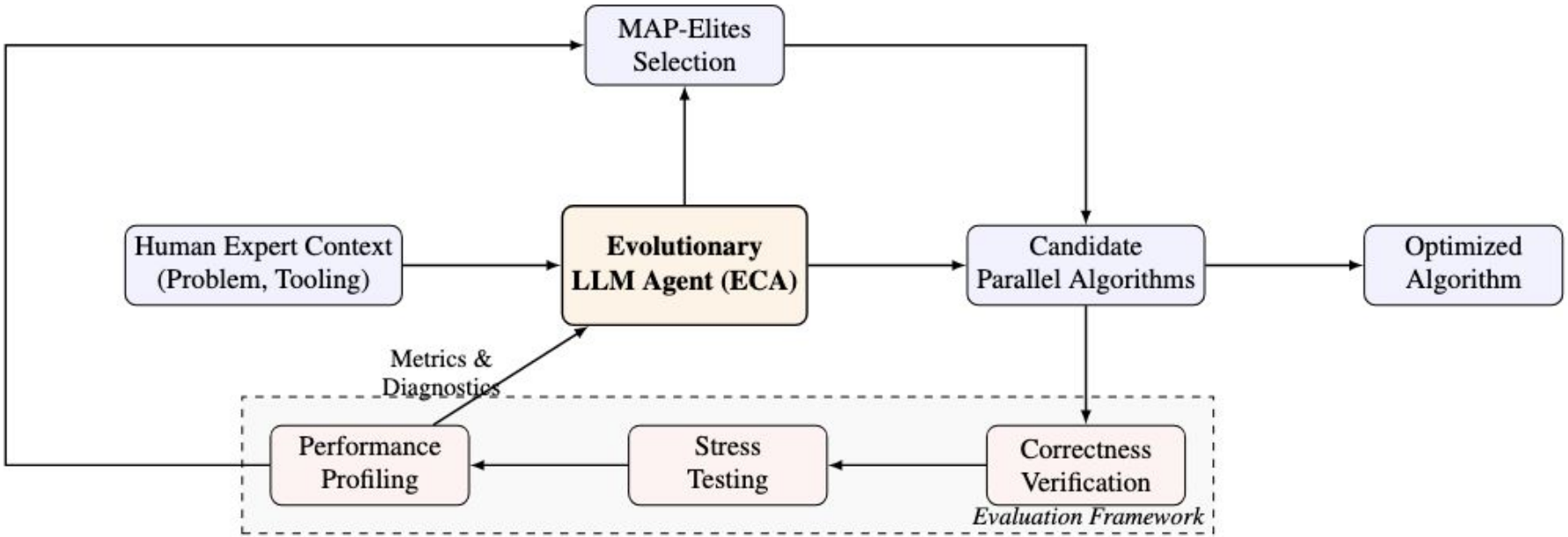
Idle threads wait on T1 → no speed

Key Insight



Functional, token-local transforms — correct by construction,
scheduling is the library's job, not the model's

The ParEVO System



Results: orders-of-magnitude speedups

107×
average speedup on ParEval
(Gemini-2.5-Parlay)

6.7B
smallest model (DeepSeek-Parlay)
beats Gemini-3-Pro

4.1×
over expert human baseline
on Maximal Independent Set (Rust)

ParEval performance (temperature = 0.2, ParlayLib)

Model	Build@1	Pass@1	Speedup@1 (AM)
GPT-5 Thinking	0.93	0.05	0.43
Gemini-3-Pro	0.25	0.23	3.68
DeepSeek-Parlay (ours, 6.7B)	**0.81**	**0.26**	**126.16**
Gemini-2.5-Parlay (ours)	**0.81**	**0.58**	**107.43**

A discovered algorithm

On BFS, ParEVO rewrote the baseline into a **direction-optimizing (Ligra-style)** traversal that the baseline never used — switching between push (sparse) and pull (dense) phases by frontier density.

Matches or exceeds expert human baselines (PBBS, RPB).

Speedup@1 is the arithmetic mean of best-performance speedups across 59 problems vs. a sequential baseline.

The correctness–speedup trade-off

An “alignment tax”: fine-tuning makes code safer, at the cost of some peak speed.

Pass@1 (graph)

0.42 → **0.76**

Correctness rises — models enforce safe API usage.

Speedup@1 (graph)

21× → **13×**

Peak speed dips — safe primitives over risky atomics.

Why this is a feature, not a bug

- Tighter runtime variance — more predictable, reliable performance.
- Structurally fewer data races — lock-free functional primitives make races and deadlocks unlikely.
- The risky-but-fast path still exists for the base model; fine-tuning trades a little peak speed for stability.

See Table 4 and the Shortest-Path runtime histograms (Figure 14).

Takeaways

1. **Abstraction choice** determines what LLMs can parallelize — high-level functional primitives align with token-local reasoning.

2. **Execution feedback** not generation alone, closes the last mile of correctness and performance.

3. **AI-driven performance engineering** moving beyond code completion toward systems that reason about scalability and hardware.

Code & Model Links

Codebase: github.com/WildAlg/ParEVO

DeepSeek-Parlay: huggingface.co/gqggez/deepseek-parlay-6.7b

Qwen-Parlay: huggingface.co/gqggez/qwen3-30b-sft-stage2-merged