KernelEvolve: Case Studies in Automatic CUDA Kernel Optimisation for Scientific Computing

Yue Sun, Jorge Luis Galvez Vallejo, Li Wang, National Computational Infrastructure

LLM-Driven Code Optimisation

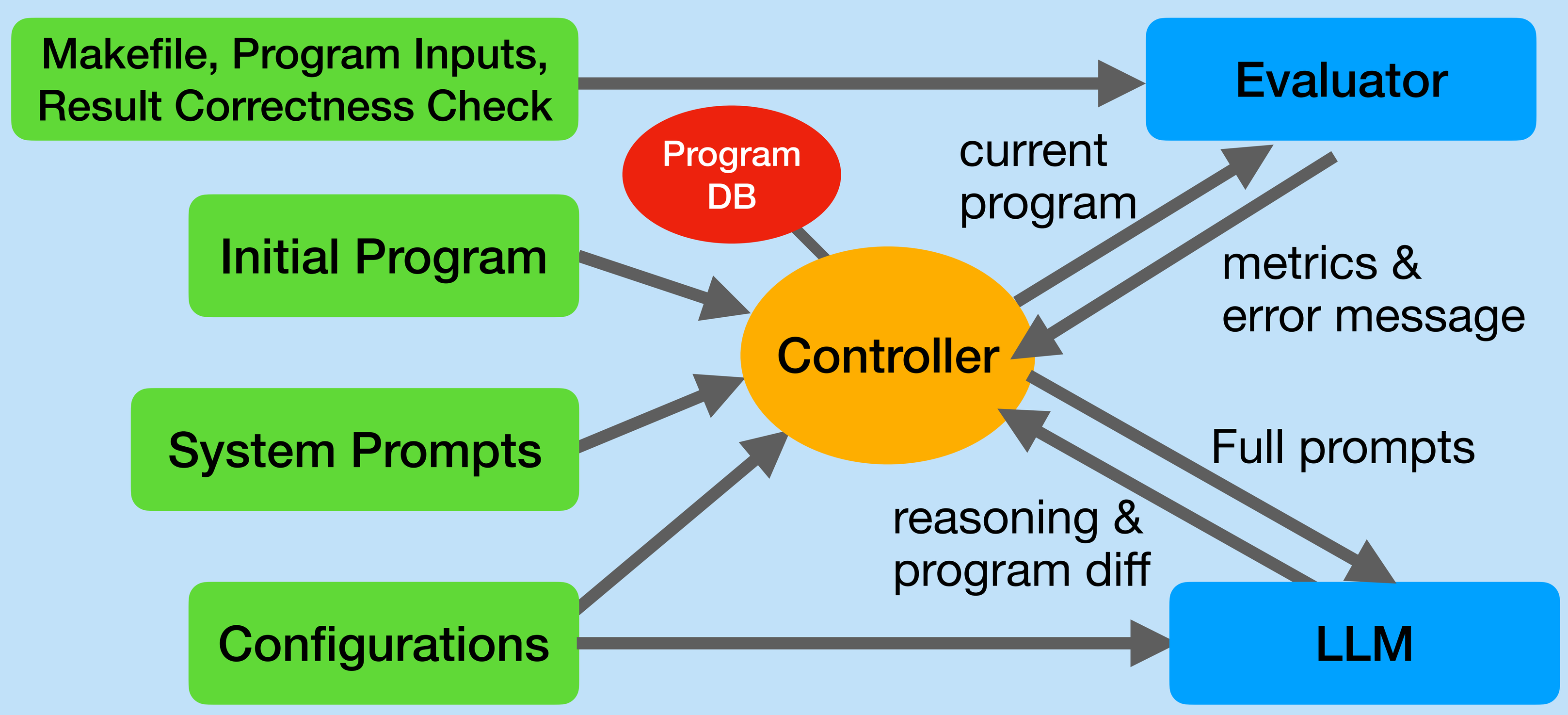
High-performance computing (HPC) relies heavily on accelerators, but optimising project-specific CUDA kernels requires deep architectural expertise and significant development time.

The Gap: While LLMs have been explored for code generation, prior work such as AI CUDA Engineer [1], KernelBench [2], and NVIDIA's attention kernel optimisation [3] focus on Deep Learning kernels and often reports performance slowdowns or instability.

Our Solution: We introduce ***KernelEvolve***, a workflow using the genetic algorithm approach AlphaEvolve [4] and its open-source implementation OpenEvolve [5] that leverages LLMs for stable, iterative CUDA optimisation targeting scientific workloads.

The ***KernelEvolve*** Workflow

It automates the optimisation loop following the AlphaEvolve algorithm and the OpenEvolve implementation, powered by LLMs (Gemini-2.5-Flash/Pro).



Minimal User Inputs: Initial Program, Makefile, Program Input, Result Correctness Check

LLM Configuration:

- Randomly choose from Flash(0.6) and Pro(0.4) LLM.
- Temperature=0.7 and P=0.95 for controlled creativity throughout the optimisation exploration

Prompts:

- **System**
  - Role: "Expert CUDA GPU programmer specialising in NVIDIA V100".
  - Safety Constraints:

```
[x] MUST NOT CHANGE: Kernel signature, input/output types, launch mapping.
[o] ALLOWED TO OPTIMIZE: Memory patterns, shared memory, vectorization,
thread block shape.
…
```

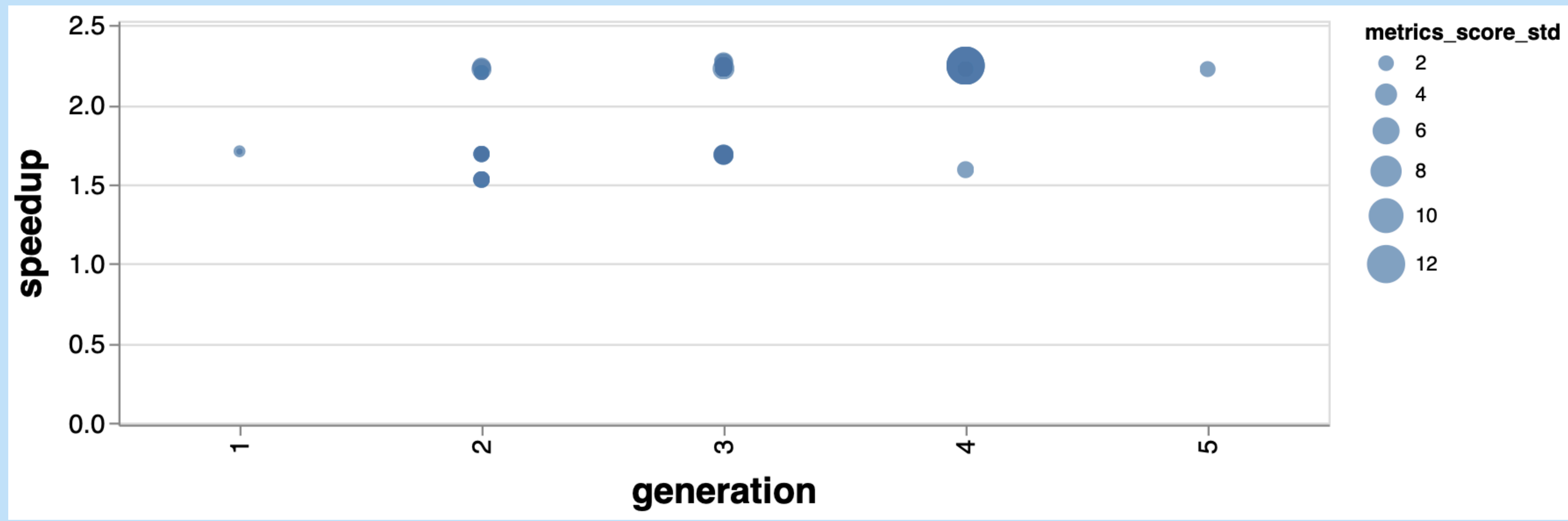
  - Success Criteria:
    - Compilation: CUDA kernel must compile without errors.
    - Correctness: Output must match baseline kernel for all valid input.
    - Performance: 5-15% improvement in effective performance score, find best trade-off between compute throughput and memory throughput.
    - Memory: No excess or wasted memory; avoid dynamic allocations in kernel.
    - Stability: No crashes, no out-of-bounds, while maintaining numerical accuracy.
- **User**: evaluation metrics, error messages included to allow feedback in the loop

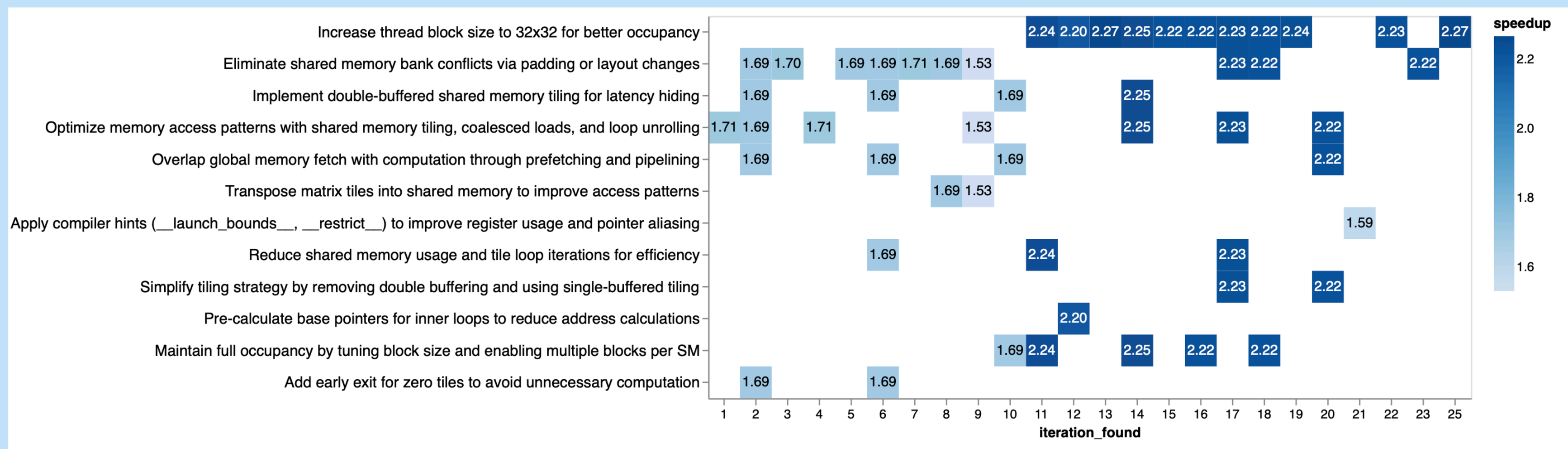
Evaluator:

- Correctness checks: compilation, runtime, and result validation
- Mean execution time measurement
- NVIDIA Nsight Compute (NCU) profiling to capture compute and memory throughput

Case Studies & Results

- Matrix Transpose
  - achieved 643 GB/s throughput
  - reward hacking"copy" strategy found for symmetric matrices, reaching 87% of cuBLAS performance
- Matrix Multiplication
  - achieved 9.5 TFLOP/s
  - reached 72% of cuBLAS performance.
  - **zero slowdowns** observed during evolution (figures below show evolution trend)
- N-body Solver
  - achieved 8.5 TFLOP/s (54% of peak FP32 performance)
  - Use of fused multiple add (FMA) intrinsics
  - Increase instruction level parallelism
  - two generated programs failed to compile
  - **single slowdown** (0.899x) at iteration 2 in generation 1
- Linear Equation Solver
  - achieved 13.6 TFLOP/s (87% of peak FP32 performance)
  - redesigned work distribution
  - applied dynamic grid sizing
  - **single slowdown** (0.998x) observed in generation 2





Reproducibility

- https://github.com/einzigsue/KernelEvolveResult

Future Works

- Work on programs in other languages such as gFortran
- Work on other GPUs such as H200 and MI250X
- Explore more LLMs
- Initialise from host(CPU) code to automatically translate to device(GPU) code

References:

[1] Robert Tjarko Lange, Aaditya Prasad, Qi Sun, Maxence Faldor, Yujin Tang, and David Ha. 2025. The AI CUDA Engineer: Agentic CUDA Kernel Discovery, optimisation and Composition. Sakana AI. Retrieved from https://pub.sakana.ai/static/paper.pdf

[2] Ouyang, A., Guo, S., Arora, S., Zhang, A. L., Hu, W., Ré, C., and Mirhoseini, A. 2025. KernelBench: Can LLMs Write Efficient GPU Kernels?. In Proceedings of the Forty-second International Conference on Machine Learning (ICML '25). Available at: https://openreview.net/forum?id=yeoN1iQT1x

[3] Terry Chen, Bing Xu, and Kirthi Devleker. Automating gpu kernel generation with deepseek-r1 and inference-time scaling. https://developer.nvidia.com/blog/automating-gpu-kernel-generation-with-deepseek-r1-and-inference-time-scaling/, February 2025

[4] Novikov, A., V˜ u, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2506.13131. Available at: https://arxiv.org/abs/2506.13131

[5] Asankhaya Sharma. 2025. OpenEvolve: an open-source evolutionary coding agent. GitHub. Retrieved from https://github.com/codelion/openevolve