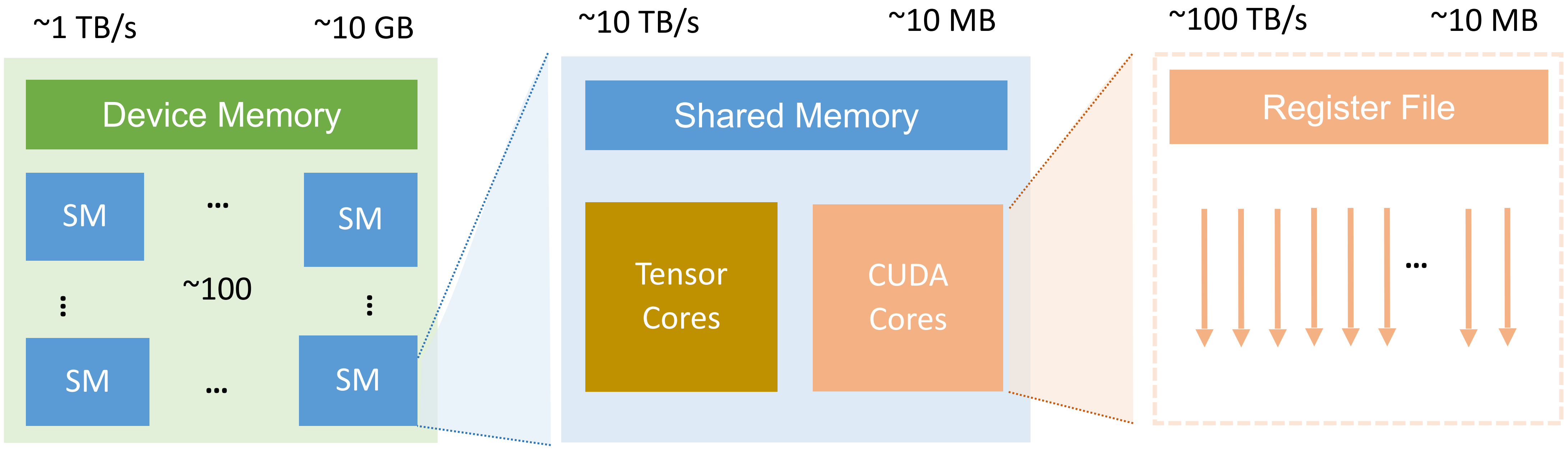
Mirage: A Multi-Level Superoptimizer for Tensor Programs

Mengdi Wu Xinhao Cheng Shengyu Liu Chunan Shi Jianan Ji

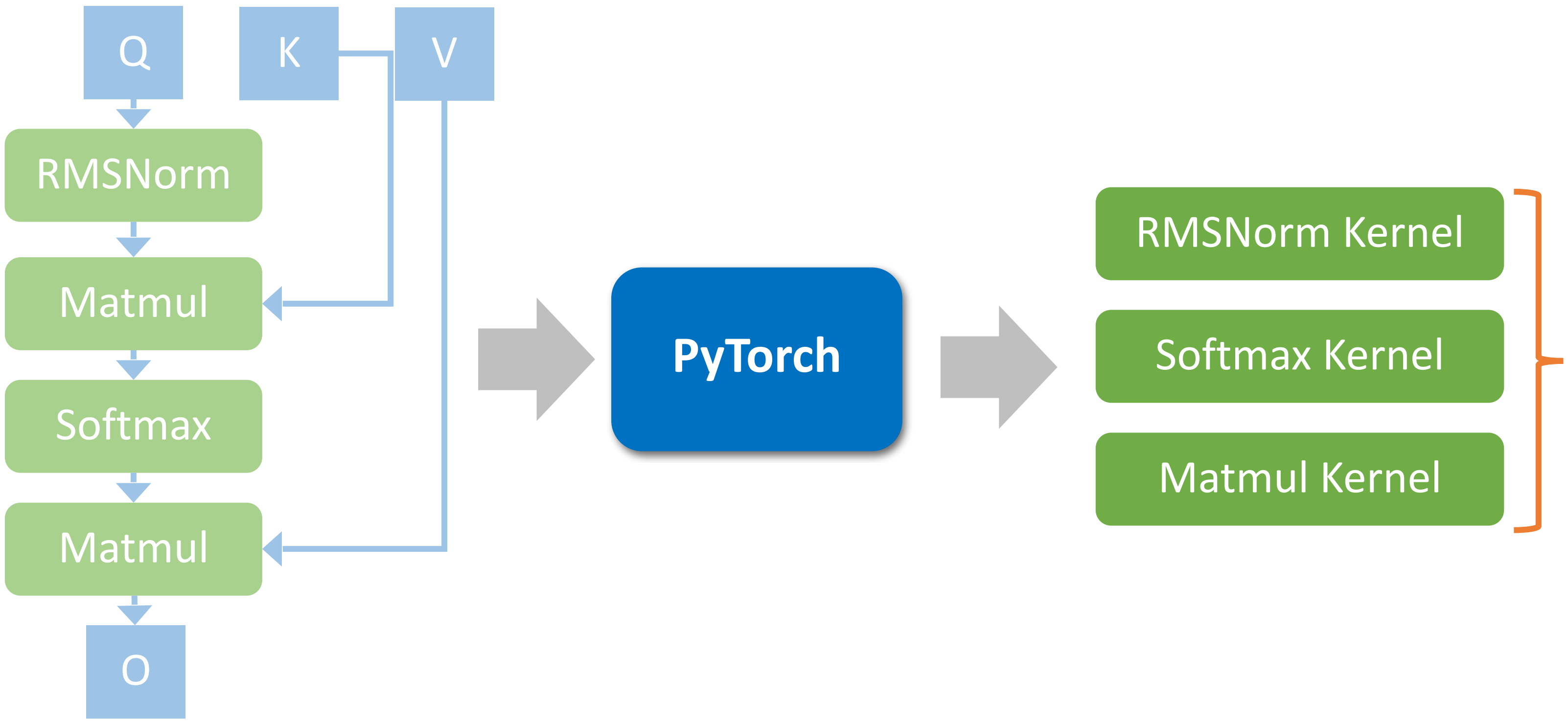
Man Kit Ao Praveen Velliengiri Xupeng Miao Oded Padon Zhihao Jia

GPUs: Complex Computation and Memory Hierarchy



Hard to design high-performance GPU kernels

Existing Systems Launch Kernels for Individual Operators

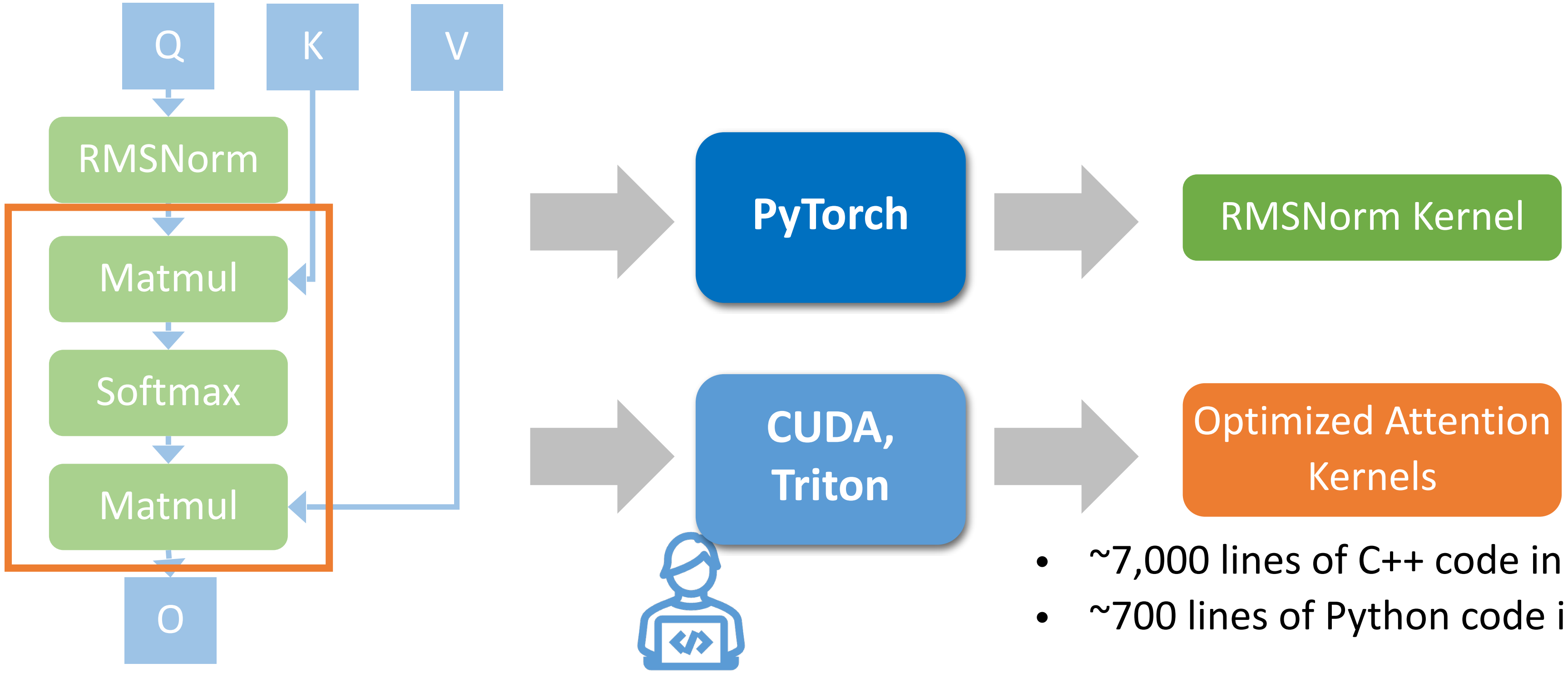


Suboptimal performance due to missing optimizations:

1. Kernel fusion
2. Algebraic transformations
3. Local memory reuse

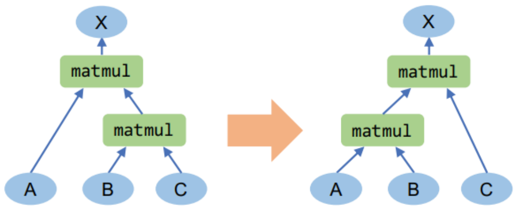
…

Manually Implement Optimized Kernels for Certain Computation Patterns



- ~7,000 lines of C++ code in CUDA
- ~700 lines of Python code in Triton

Mirage: A *Multi-Level* Superoptimizer



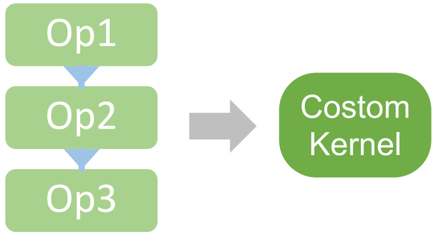
Algebraic Transformation

(TASO[SOSP'19], PET[OSDI'21],...)

```
for yo in range(128):
  for xo in range(128):
    C[yo*8:yo*8+8][xo*8:xo*8+8] = 0
    for ko in range(128):
      for yi in range(8):
        for xi in range(8):
          for ki in range(8):
            C[yo*8+yi][xo*8+xi] +=
               A[ko*8+ki][yo*8+yi] * B[ko*8+ki][xo*8+xi]
```

Schedule Transformation

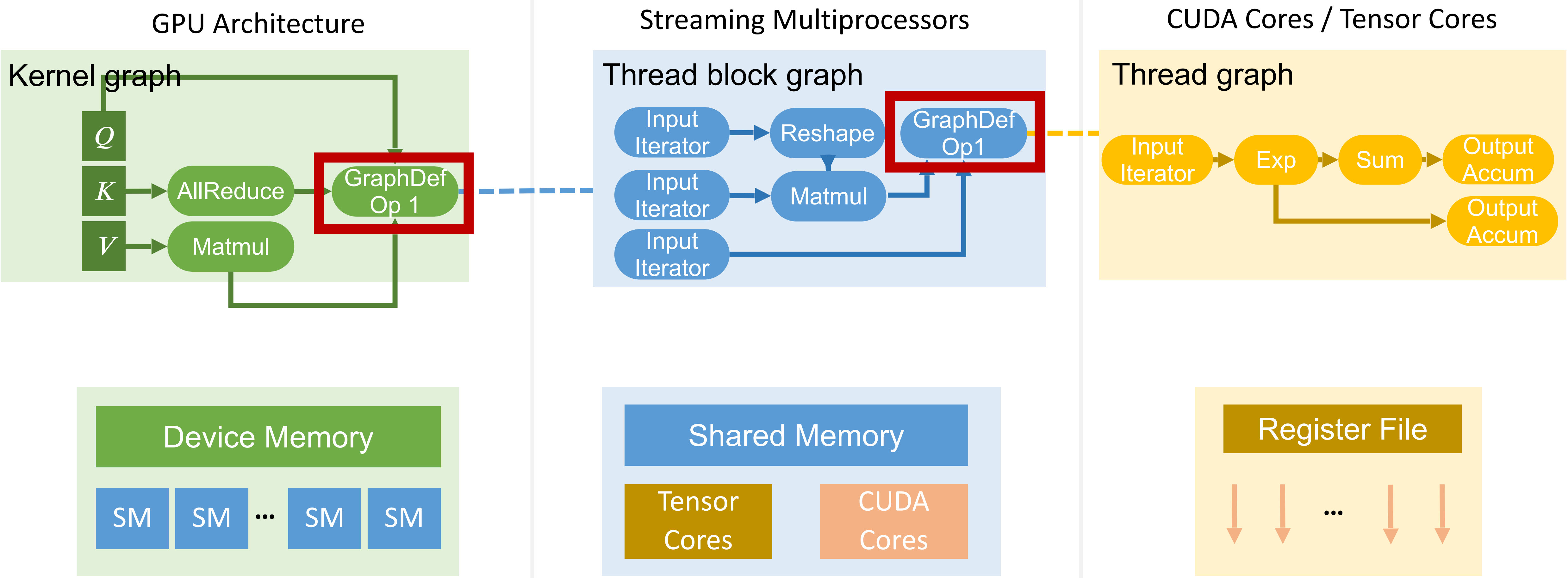
(TVM[OSDI'18], Ansor[OSDI'20],...)



Custom Kernel Generation

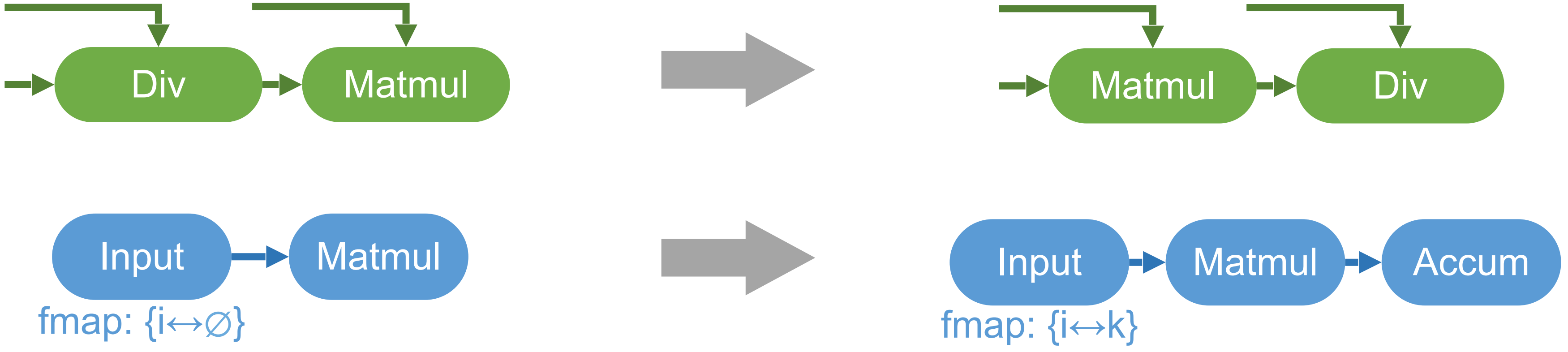
- **Less engineering effort**: 7,000 lines of CUDA code / 700 lines of Triton code → a few lines of Python code in Mirage
- **Better performance**: outperform existing systems by up to 3.3x
- **Easy adaptation**: do not rely on manual implementation

μGraph: Multi-Level Graph Representation



Challenge: Search Space Exploration

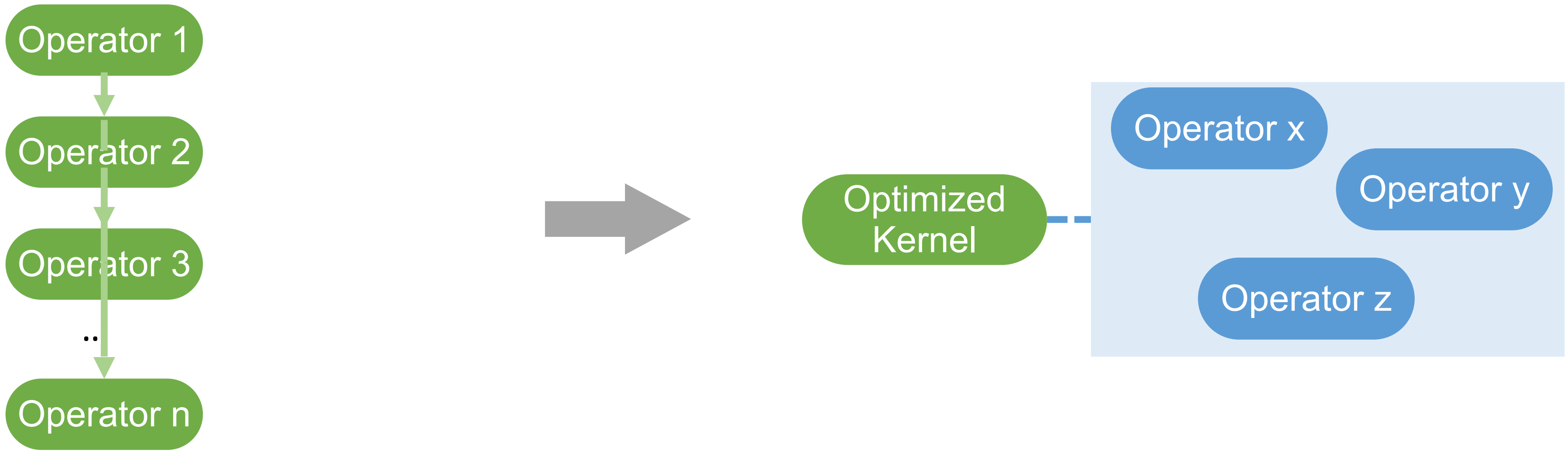
Transformation-based approach: define *transformation rules* $\mathscr{T}: G_{pattern} \mapsto G_{opt}$



Easy for optimizations within one level,
(Transformations can be summarized as *limited* rules)
but …

Challenge: Search Space Exploration

Transformation-based approach: define *transformation rules* $\mathscr{T} : G_{pattern} \mapsto G_{opt}$

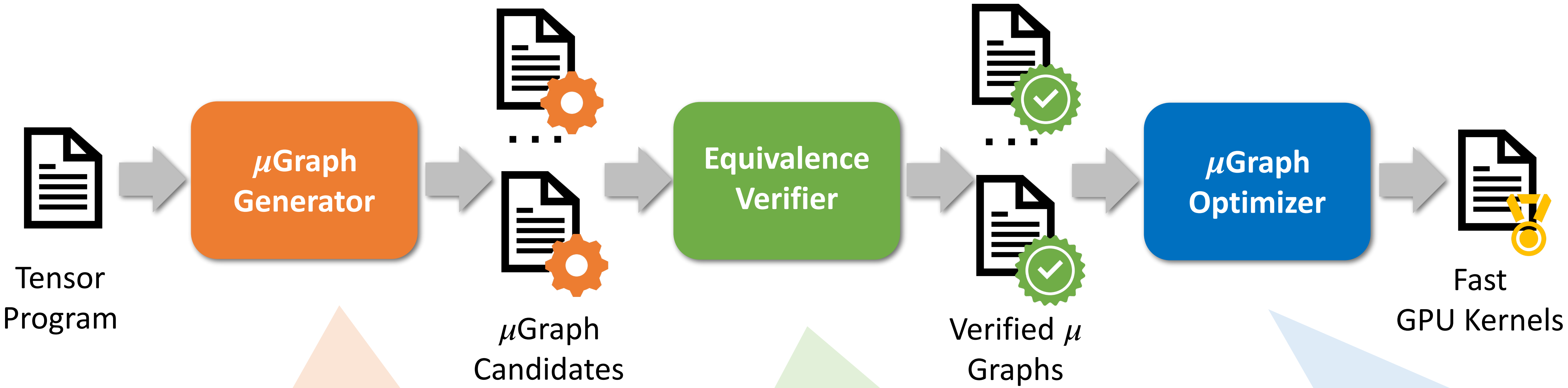


Infinitely many possible patterns

No common rules for optimized kernels

Hard for multi-level optimizations
(Transformations are *dense* and *irregular* functions)

Mirage Overview



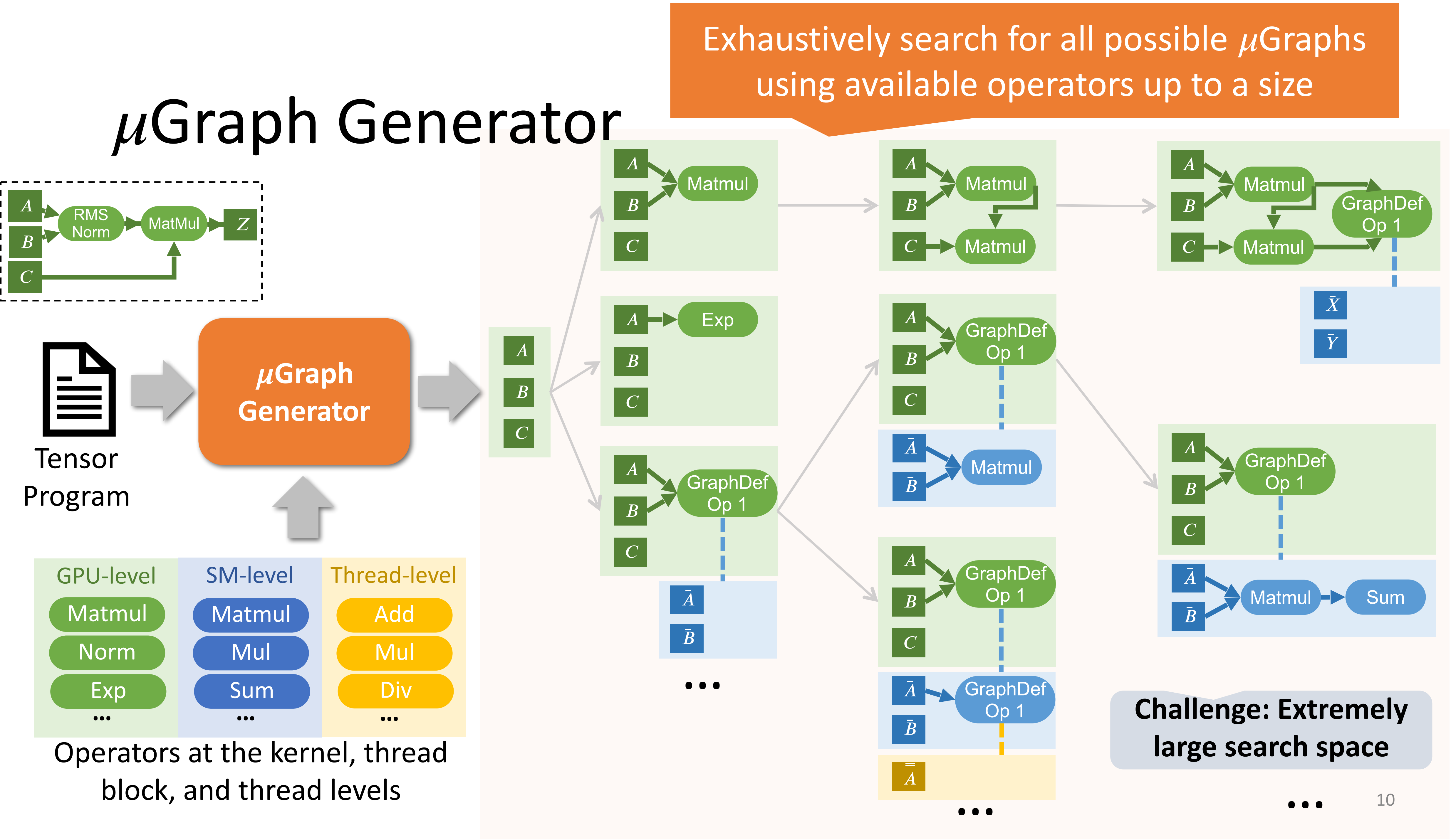
Generate all possible μ Graphs up to a bounded size using **exhaustive search**

Check whether generated μGraphs are correct by **random testing** with **theoretical guarantee**

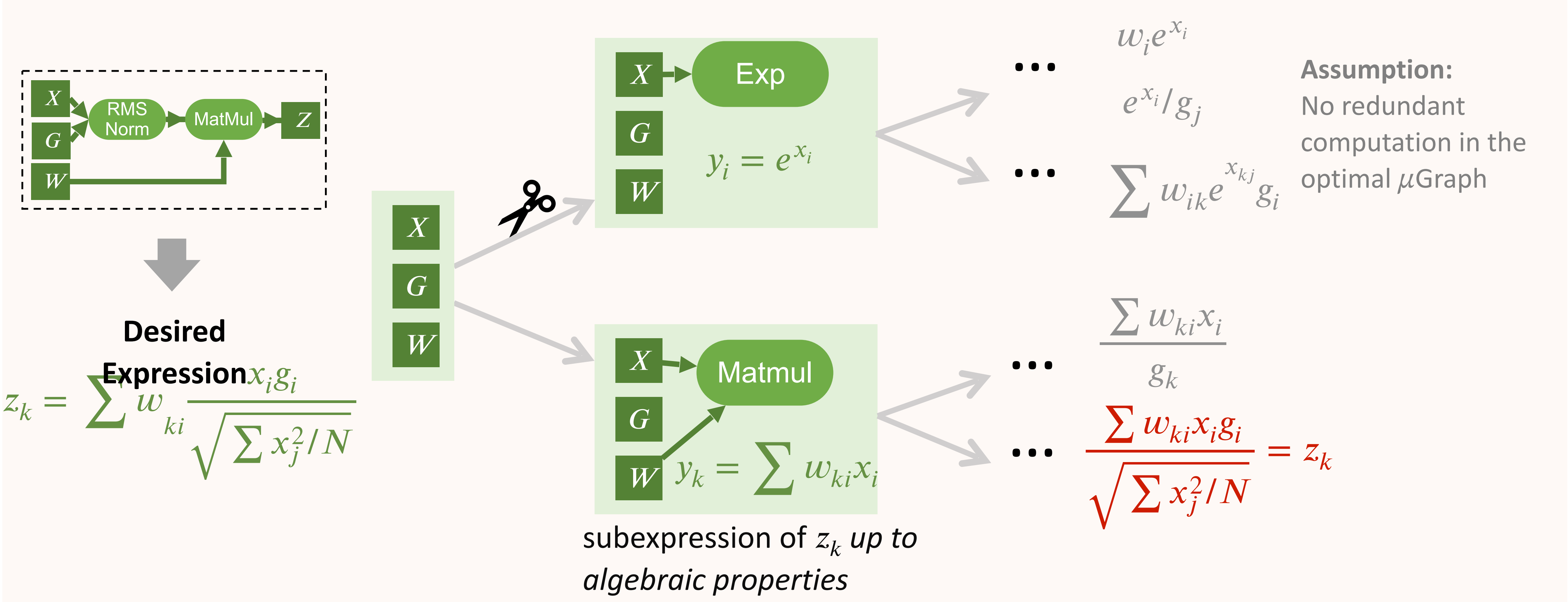
Optimizations that do not affect correctness of μGraphs:

- Tensor layouts
- Memory planning
- Operator scheduling

μGraph Generator



Expression-Guided Pruning



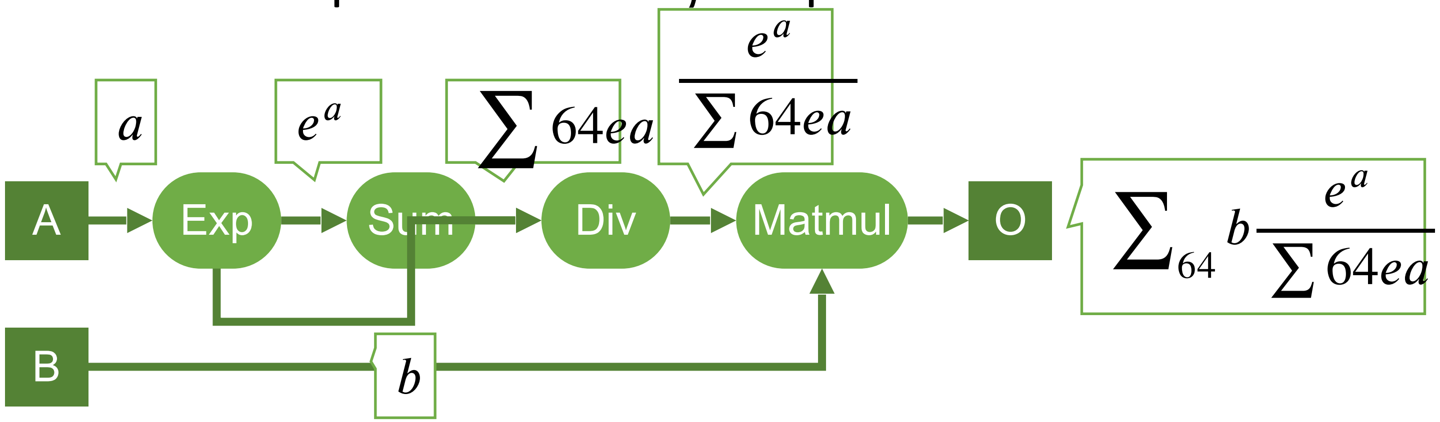
Full-information expressions are too complicated to reason about

Abstract Expression

Key idea: abstract away the index details

- E.g., $C = A \times B$: $c_{ij} = \sum_{k=1}^{64} a_{ik} b_{kj} \Rightarrow$ abstract expression is $\sum_{64} ab$

Recursively compute abstract expressions for a μGraph



- **Capture most semantic information**
- **Easy to reason about**

Axioms for Abstract Expression

Mirage uses first-order logic to reason about abstract expression relations

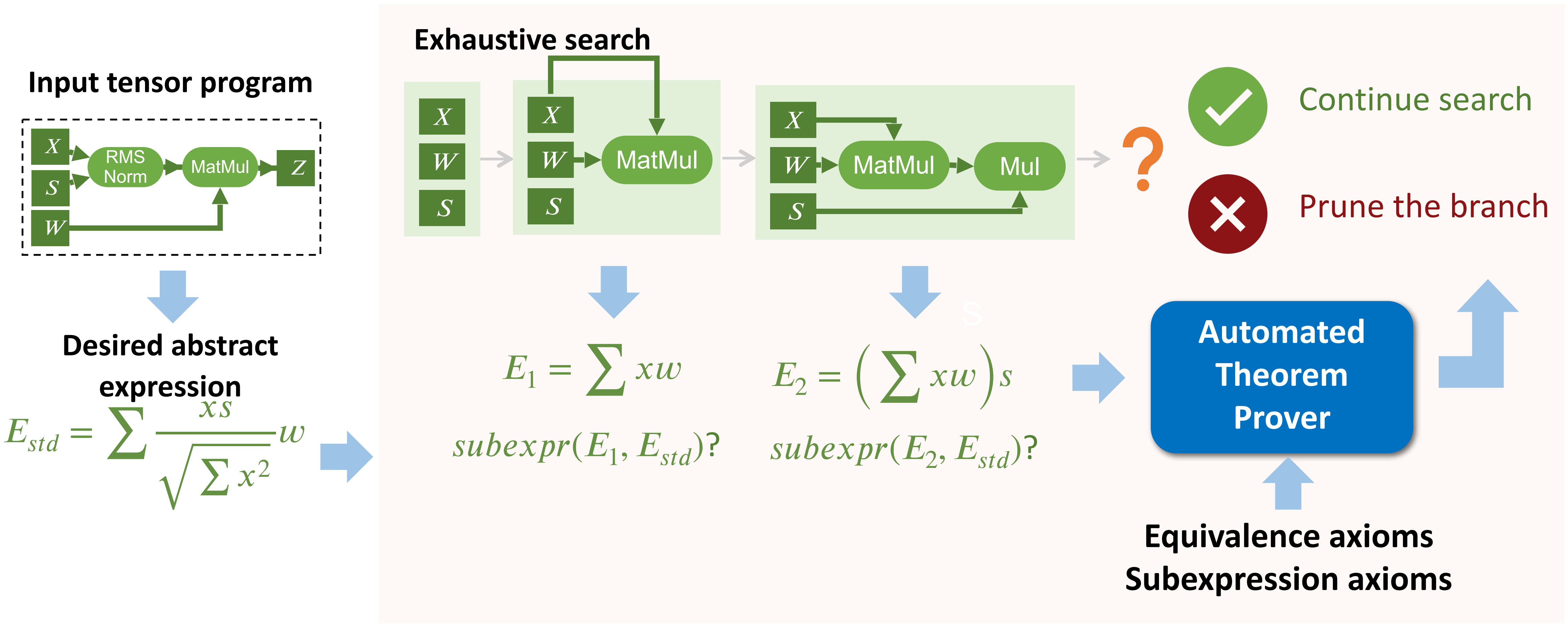
Equivalence axioms

$\forall x, y.\ \mathsf{add}(x, y) = \mathsf{add}(y, x)$	commutativity
$\forall x, y.\ \mathsf{mul}(x, y) = \mathsf{mul}(y, x)$	commutativity
$\forall x, y, z.\ \mathsf{add}(x, \mathsf{add}(y, z)) = \mathsf{add}(\mathsf{add}(x, y), z)$	associativity
$\forall x, y, z.\ \mathsf{mul}(x, \mathsf{mul}(y, z)) = \mathsf{mul}(\mathsf{mul}(x, y), z)$	associativity
$\forall x, y, z.\ \mathsf{add}(\mathsf{mul}(x, z), \mathsf{mul}(y, z)) = \mathsf{mul}(\mathsf{add}(x, y), z)$	distributivity
$\forall x, y, z.\ \mathsf{add}(\mathsf{div}(x, z), \mathsf{div}(y, z)) = \mathsf{div}(\mathsf{add}(x, y), z)$	associativity
$\forall x, y, z.\ \mathsf{mul}(x, \mathsf{div}(y, z)) = \mathsf{div}(\mathsf{mul}(x, y), z)$	associativity
$\forall x, y, z.\ \mathsf{div}(\mathsf{div}(x, y), z) = \mathsf{div}(x, \mathsf{mul}(y, z))$	associativity
$\forall x.\ x = \mathsf{sum}(1, x)$	identity reduction
$\forall x, i, j.\ \mathsf{sum}(i, \mathsf{sum}(j, x)) = \mathsf{sum}(i * j, x)$	associativity
$\forall x, y, i.\ \mathsf{sum}(i, \mathsf{add}(x, y)) = \mathsf{add}(\mathsf{sum}(i, x), \mathsf{sum}(i, y))$	associativity
$\forall x, y, i.\ \mathsf{sum}(i, \mathsf{mul}(x, y)) = \mathsf{mul}(\mathsf{sum}(i, x), y)$	distributivity
$\forall x, y, i.\ \mathsf{sum}(i, \mathsf{div}(x, y)) = \mathsf{div}(\mathsf{sum}(i, x), y)$	distributivity

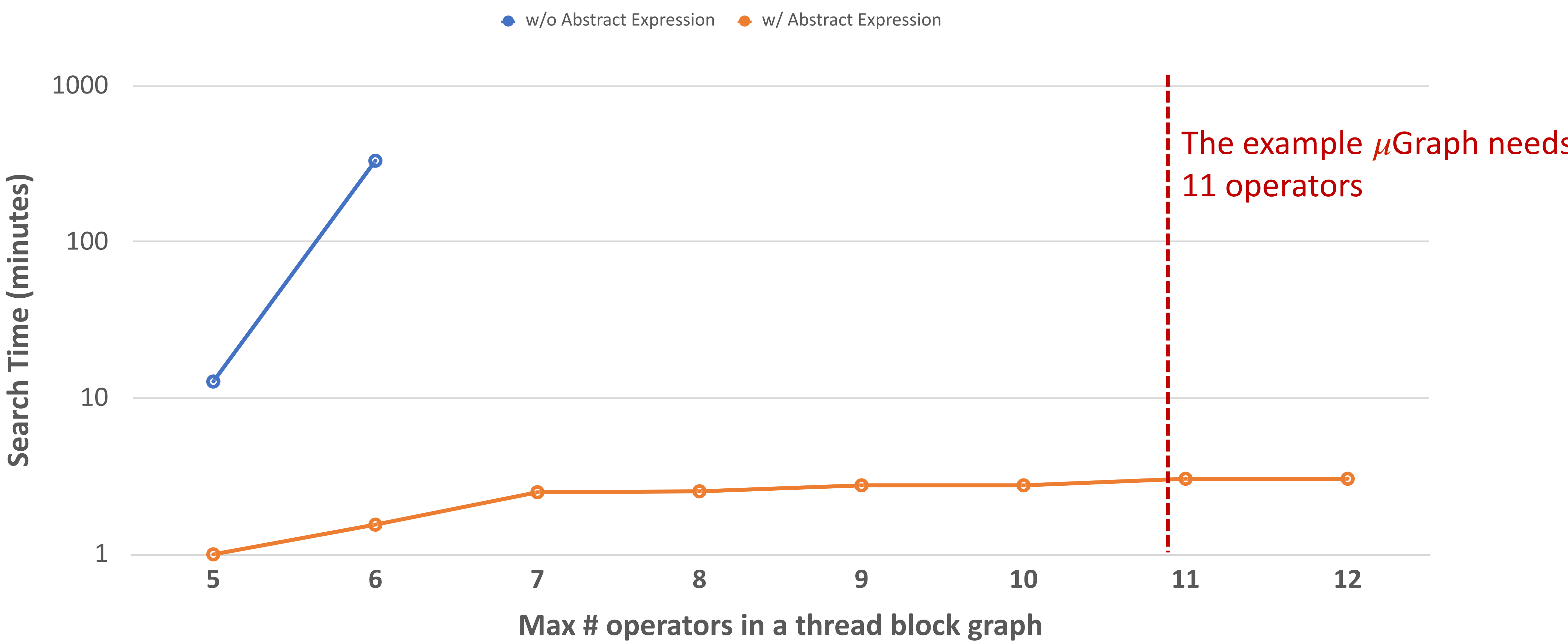
Subexpression axioms

$\forall x, y.\ \mathsf{subexpr}(x, \mathsf{add}(x, y))$	
$\forall x, y.\ \mathsf{subexpr}(x, \mathsf{mul}(x, y))$	
$\forall x, y.\ \mathsf{subexpr}(x, \mathsf{div}(x, y))$	
$\forall x, y.\ \mathsf{subexpr}(y, \mathsf{div}(x, y))$	
$\forall x.\ \mathsf{subexpr}(x, \mathsf{exp}(x))$	
$\forall x, i.\ \mathsf{subexpr}(x, \mathsf{sum}(i, x))$	
$\forall x.\ \mathsf{subexpr}(x, x)$	reflexivity
$\forall x, y, z.\ \mathsf{subexpr}(x, y) \wedge \mathsf{subexpr}(y, z) \rightarrow \mathsf{subexpr}(x, z)$	transitivity

Abstract Expression-Guided Search

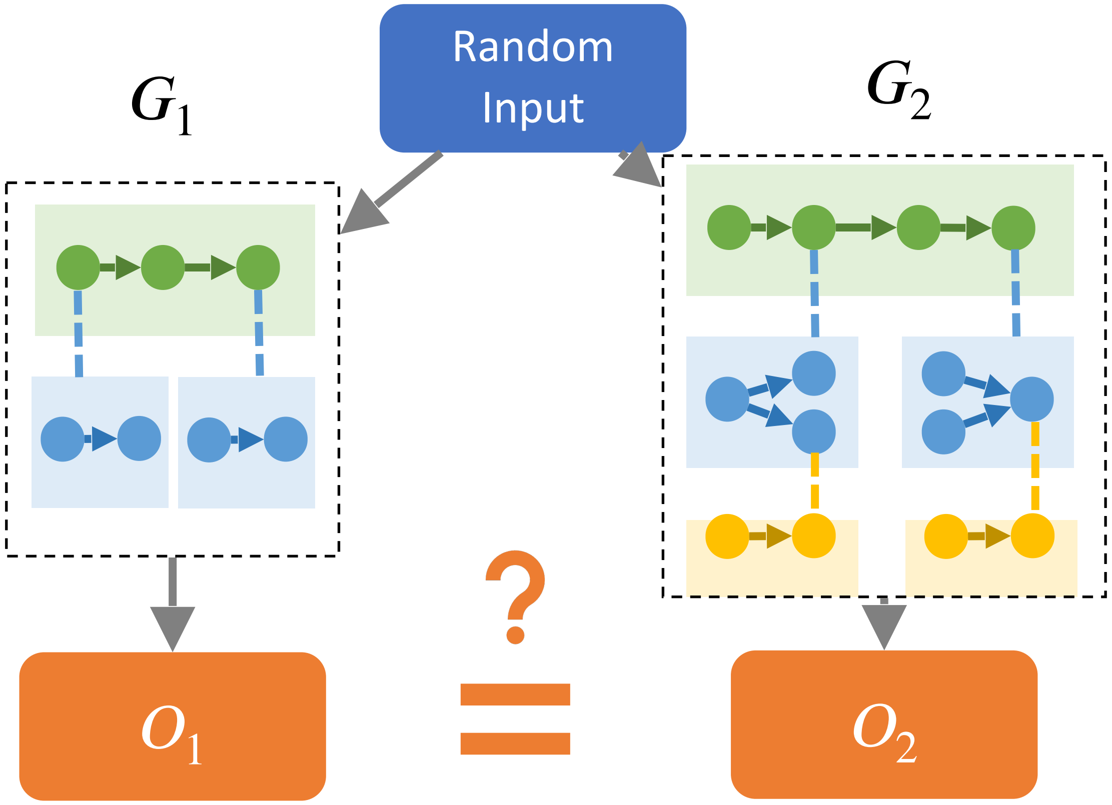


Abstract Expression Significantly Improves Scalability



Probabilistic Equivalence Verifier

Idea: use random inputs in ***finite fields*** to examine μGraph equivalence

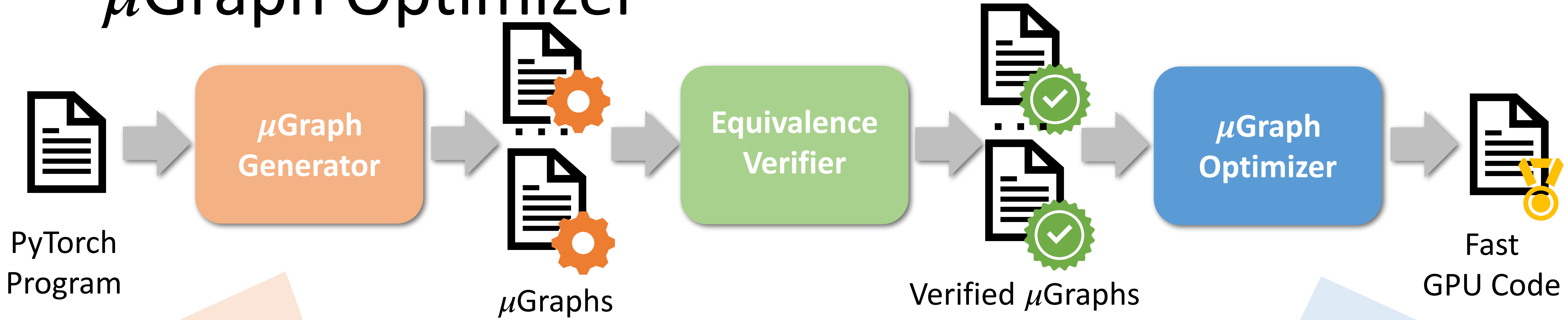


Theorem 1: if G_1 is equivalent to G_2, then $O_1 = O_2$

Theorem 2: if G_1 is not equivalent to G_2, then $O_1 \neq O_2$ with a certain probability p^*

* $p \geq \frac{1}{T}$, where T is the size of intermediate tensors

μGraph Optimizer



Only consider output-alternating optimizations:

- Algebraic transformations
- Kernel instantiation
- Compute organization

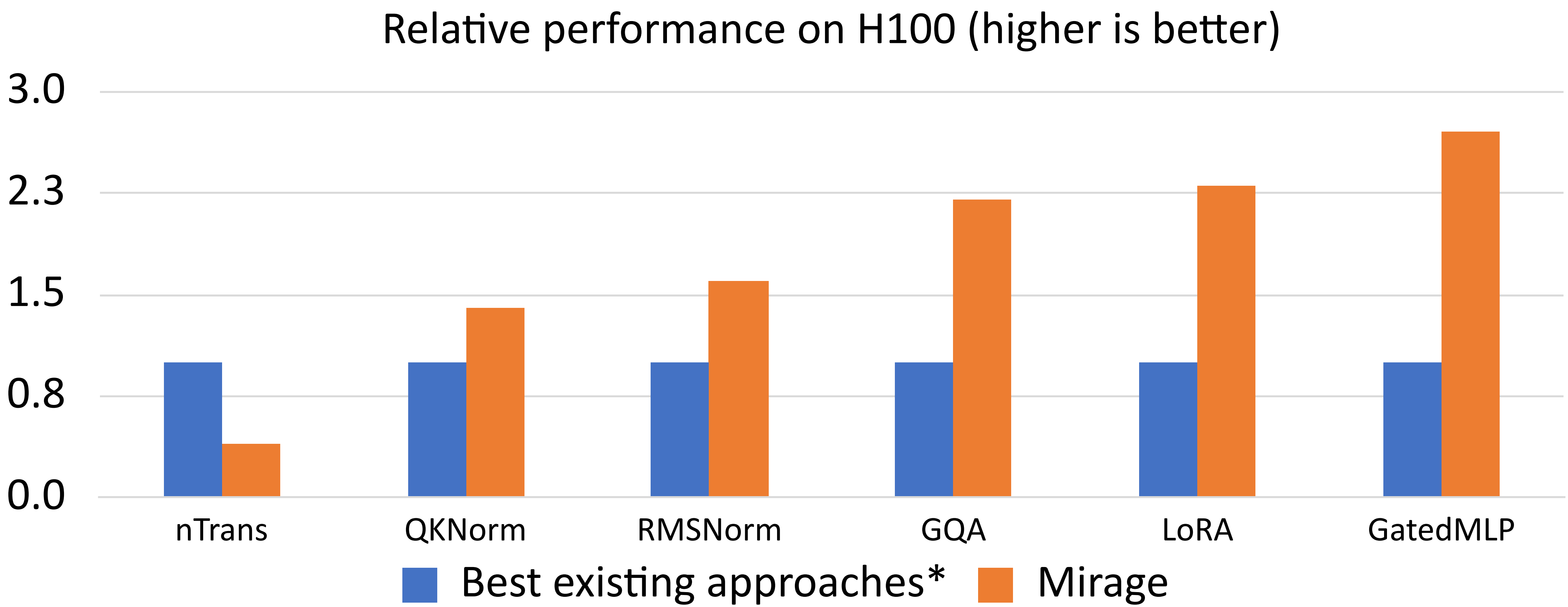
Reduce generator's search space

Other optimizations are deferred to μ Graph Optimizer:

- Tensor layouts
- Memory planning
- Operator scheduling

Solve these tasks optimally

Mirage Outperforms Existing Approaches

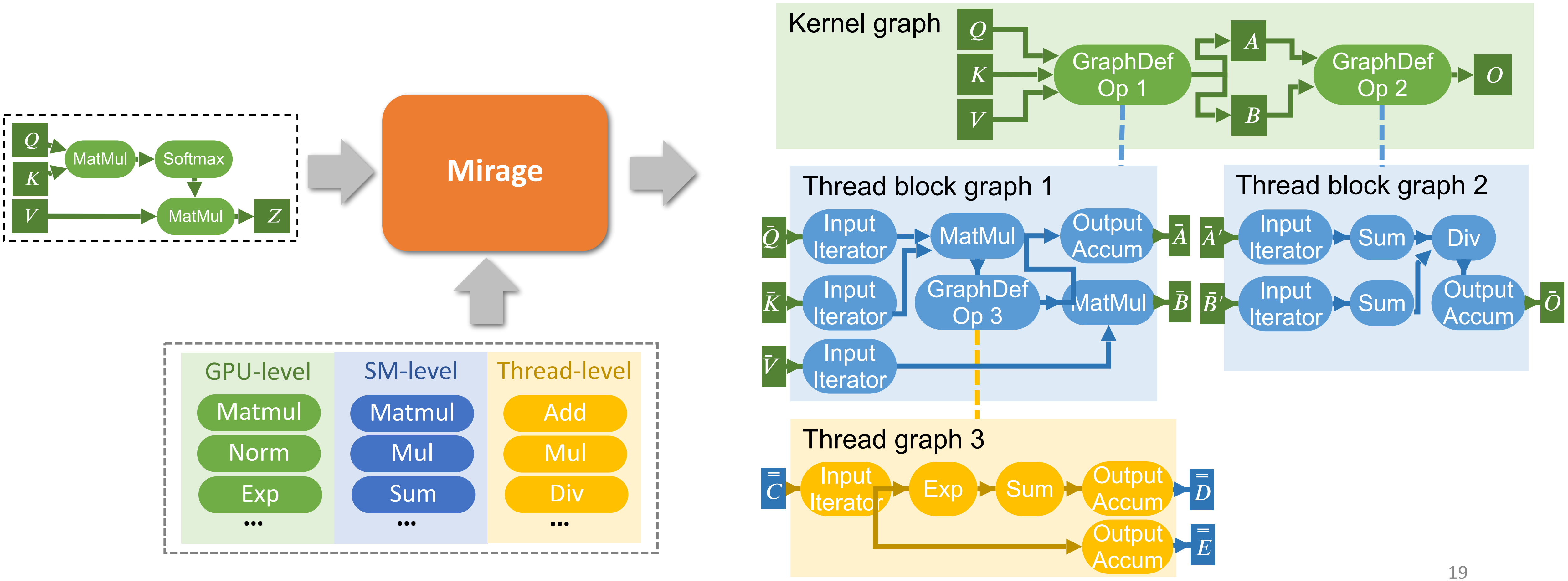


Vender-provided
Expert-written
Compiler-generated

*Best existing approaches are the best of cuDNN/cuBLAS, FlashAttention, PyTorch, TensorRT, Triton, and TVM.

Mirage Discovers Hardware-Customized μGraphs

Find μGraphs similar to expert-written implementations* for attention on A100

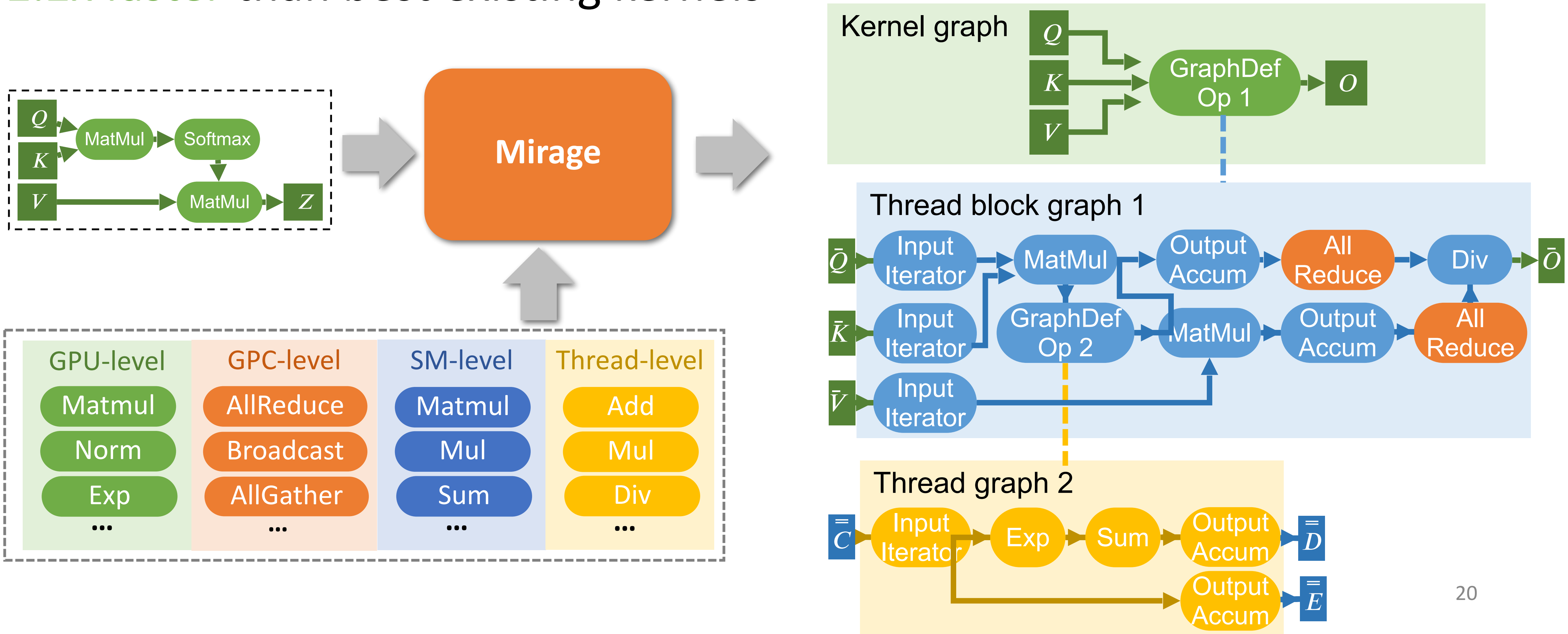


* FlashDecoding++: Faster LLM Inference with Asynchronization, Flat GEMM Optimization, and Heuristics. MLSys'24

Mirage Discovers Hardware-Customized μGraphs

Leverage GPC-level AllReduce to accelerate attention on H100

- 2.2x faster than best existing kernels



Mirage: A Multi-Level Superoptimizer

- *Algebraic transformation + Schedule transformation + New kernel generation*
- **Minimal engineering effort:** A few lines of Python codes from users
- **High performance:** Outperform existing systems by up to 3.3x

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Our Paper @ OSDI 2025

https://www.usenix.org/conference/osdi25/presentation/wu-mengdi

Our Repo

https://github.com/mirage-project/mirage