SIP: Autotuning GPU Native Schedules via Stochastic Instruction Perturbation

- Guoliang He, Eiko Yoneki
- University of Cambridge, UK

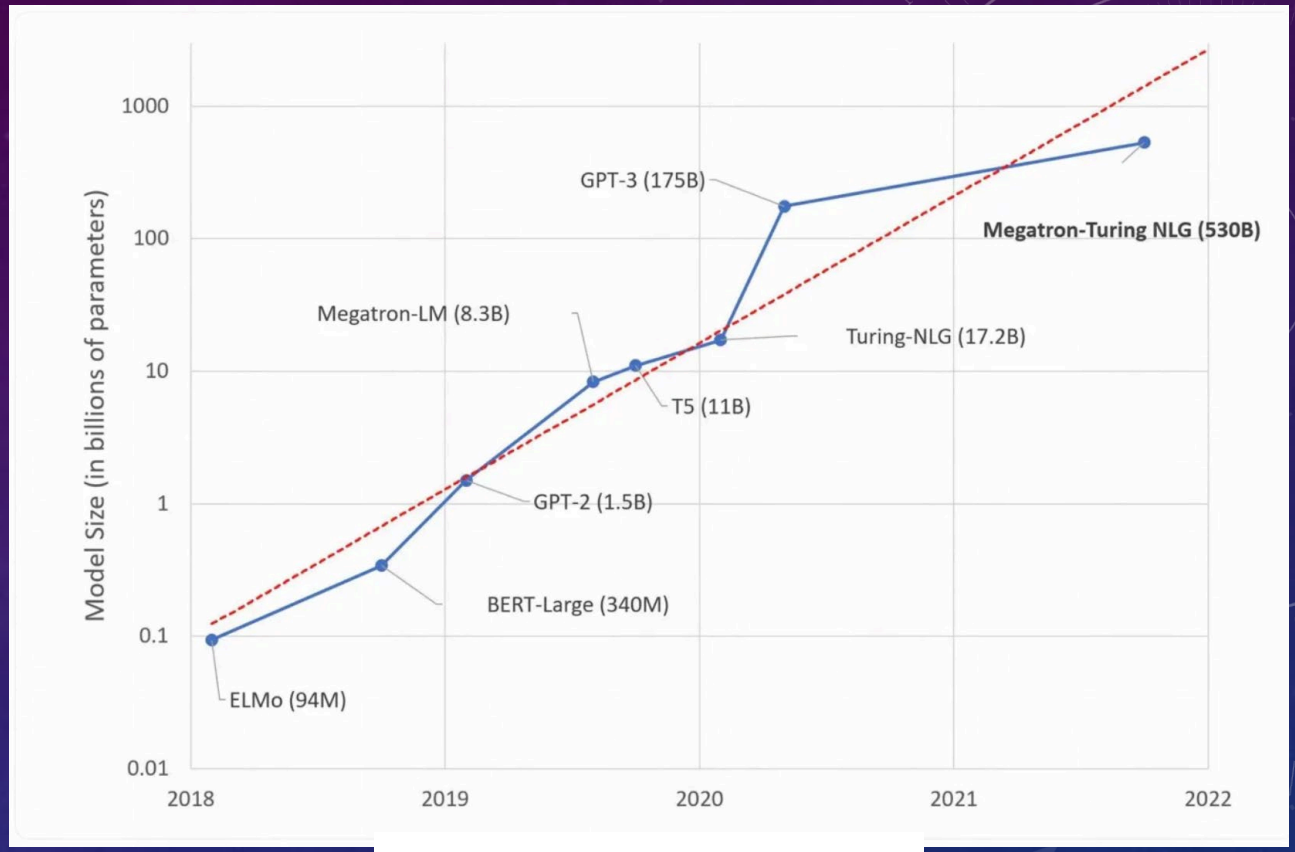

```
1   LDGSTS .E.BYPASS.128 [R219+0x4000], desc[UR16][
      R10.64], P0 ;
2   IMAD .WIDE R18, R9, 0x80, R10 ;
3   LDGSTS .E.BYPASS.128 [R219+0x4800], desc[UR16][
      R44.64], P0 ;
4   IMAD .WIDE.U32 R16, R222, 0x2, R64 ;
5   LDGSTS .E.BYPASS.128 [R219+0x5000], desc[UR16][
      R46.64], P0 ;
6   IMAD .WIDE.U32 R10, R222, 0x2, R60 ;
7   LDGSTS .E.BYPASS.128 [R219+0x5800], desc[UR16][
      R48.64], P0 ;
8   MOV R33, c[0x0][0x1b0] ;
9   LDGDEPBAR ;
10
```

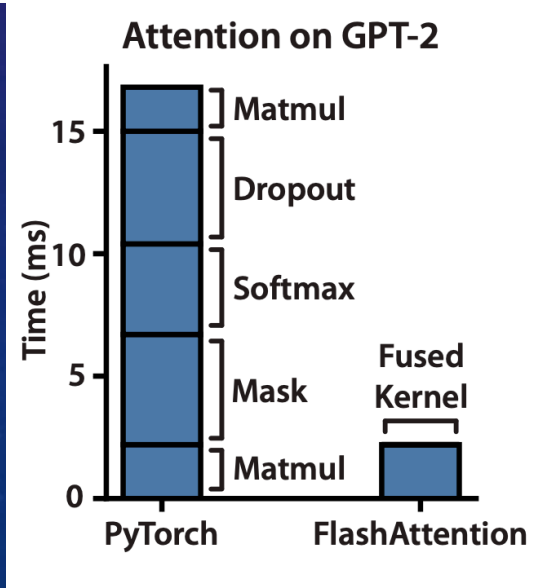
SIP

```
1   IMAD .WIDE R18, R9, 0x80, R10 ;
2   LDGSTS .E.BYPASS.128 [R219+0x4000], desc[UR16][
      R10.64], P0 ;
3   LDGSTS .E.BYPASS.128 [R219+0x4800], desc[UR16][
      R44.64], P0 ;
4   IMAD .WIDE.U32 R16, R222, 0x2, R64 ;
5   LDGSTS .E.BYPASS.128 [R219+0x5000], desc[UR16][
      R46.64], P0 ;
6   LDGSTS .E.BYPASS.128 [R219+0x5800], desc[UR16][
      R48.64], P0 ;
7   IMAD .WIDE.U32 R10, R222, 0x2, R60 ;
8   MOV R33, c[0x0][0x1b0] ;
9   LDGDEPBAR ;
10
```

BACKGROUND: LLMS

- LLMs are remarked by their substantial computational demands
- To fully utilize the hardware resources, ML practitioner designs customized CUDA kernels, such as Flash Attention [1]
- In this work, we explore the optimization at GPU assembly level





[1] Tri Dao, et. al. 2022.

BACKGROUND: CUDA COMPILATION

CUDA kernels compilation pipeline:

C++/CUDA → PTX → SASS → Cubin

- C++/CUDA: the common programming interface; CUDA kernels as a C++ function
- PTX: the lowest officially supported programming interface, usually embedded in CUDA/C++
- SASS: GPU native instructions. GPU architectures-dependent. (SIP optimizes at this level)
- Cubin: executable binary

BACKGROUND: GPU ASSEMBLY OPTIMIZATION

- Latency Hiding: overlap memory I/O and computation units
- GPU executes static instruction schedule
- ***Manually reordering*** the instruction schedules is performed by prior works [2]. ***Trial-and-error*** is proposed to find the optimal sequences [3]
- Observation: large amount of time and effort are needed to optimize ***one*** CUDA kernel

[2] D. Yan, et. al. 2020. [3] X. Zhang, el. al. 2017

MOTIVATIONS

Limitations:

- Manual scheduling is error-prone and requires in-depth CUDA expertise
- Cannot keep up with the development of new operators

Method:

- We aim to apply ***automatic optimization*** by defining a search space of possible instruction schedules and perform stochastic search

SIP

Search space:

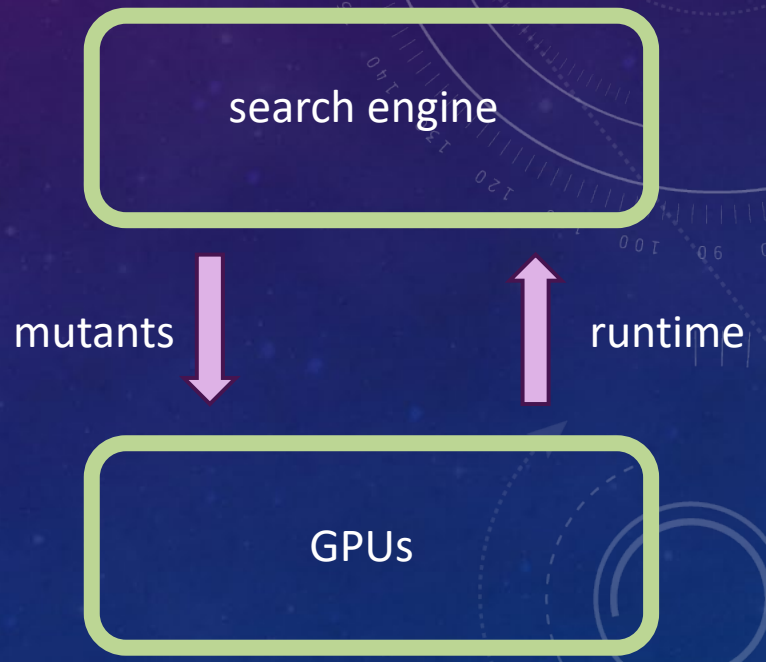
- Original: full permutation of instructions; computationally intractable
- Pruning: consider only ***global read and write*** memory instructions, e.g. LDG, STG etc

Mutation:

- Randomly select an instruction and reorder an instruction above or below

Feedback signal:

- Assemble the mutated SASS and run on GPUs



SIP

Putting together:

- Use simulated annealing to form the control loop
- Simulated annealing is a metaheuristic stochastic optimization method to explore a discrete search space

Algorithm 1 Simulated annealing for stochastic instruction Perturbation (SIP).

Initialize $T_{\max}$, $T_{\min}$, x
Initialize $x_{\text{best}} \leftarrow x$
$T \leftarrow T_{\max}$
while $T > T_{\min}$ **do**
    Generate a new schedule x' by perturbing x
    $\Delta E = \text{Energy}(x') - \text{Energy}(x)$
    if $\Delta E < 0$ **then**
        Accept x' as the current schedule: $x \leftarrow x'$
        if $\text{Energy}(x) < \text{Energy}(x_{\text{best}})$ **then**
            Update the best schedule: $x_{\text{best}} \leftarrow x$
        end if
    else
        if $r < \exp(-\Delta E/T)$ **then**
            Accept x' as the current schedule: $x \leftarrow x'$
        end if
    end if
    Cool down the system: $T \leftarrow T \times L^{-1}$
end while
return x_{best}

IMPLEMENTATION

Integrate to OpenAI Triton:

- A MLIR-based compiler to write CUDA kernels
- Pytorch 2's default backend
- 1 line of code change to use SIP

```
@triton.jit
def vector_add(x_ptr, out_ptr):
    ...
```

```
@sip.jit(ret_ptr=1)
def vector_add(x_ptr, out_ptr):
    ...
```

Workflow: search then look-up

Probabilistic Testing:

- The formal semantics of SASS is closed-source
- Probabilistic testing to evaluate end-to-end equivalency
- The compiler hint *ret_ptr* allows SIP to generate reference input/output

EVALUATIONS

- CUDA kernels: fused attention (flash-attention) and fused GEMM-leakyReLU
- GPU: NVIDIA A100 80GB
- Software: NVCC 12.2 and Triton v2.1.0
- Profiler: Nsight Compute

KERNEL THROUGHPUT

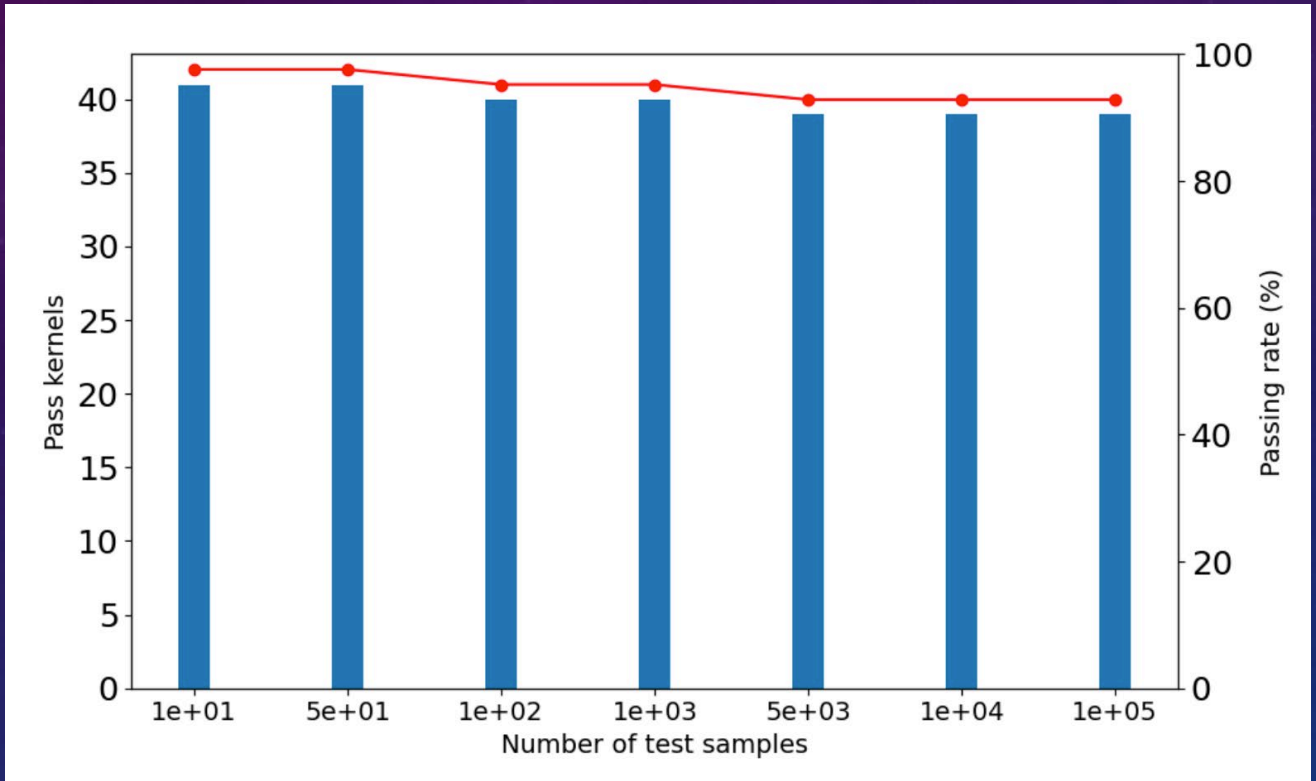
Table 2. Fused attention on A100. The input data have the format of [1, 4, 16384, 64], representing *batch size, number of heads, sequence length, head dimension.*

Metric	Unit	SIP	Triton
DRAM Frequency	cycle/ns	1.51	1.49
SM Frequency	cycle/ns	1.06	1.05
Memory Throughput	%	**33.19**	31.38
DRAM Throughput	%	1.12	1.07
Duration	ms	**1.29**	1.37
L1/TEX Cache Throughput	%	44.46	44.69
L2 Cache Throughput	%	16.29	15.41
Compute (SM) Throughput	%	**45.87**	43.39

Table 3. Fused GEMM LeakyReLU on A100. The input data have the format of [512, 512, 2048] (M, N, K).

Metric	Unit	SIP	Triton
DRAM Frequency	cycle/ns	1.50	1.50
SM Frequency	cycle/ns	1.01	1.02
Memory Throughput	%	**45.64**	40.52
DRAM Throughput	%	9.14	8.14
Duration	μs	**23.97**	26.91
L1/TEX Cache Throughput	%	38.16	37.94
L2 Cache Throughput	%	45.64	40.52
Compute (SM) Throughput	%	**19.98**	17.73

TRANSFORMATION CORRECTNESS



DISCOVERING BETTER SCHEDULES

```
1  LDGSTS .E.BYPASS.128 [R219+0x4000], desc[UR16][
     R10.64], P0 ;
2  IMAD .WIDE R18, R9, 0x80, R10 ;
3  LDGSTS .E.BYPASS.128 [R219+0x4800], desc[UR16][
     R44.64], P0 ;
4  IMAD .WIDE.U32 R16, R222, 0x2, R64 ;
5  LDGSTS .E.BYPASS.128 [R219+0x5000], desc[UR16][
     R46.64], P0 ;
6  IMAD .WIDE.U32 R10, R222, 0x2, R60 ;
7  LDGSTS .E.BYPASS.128 [R219+0x5800], desc[UR16][
     R48.64], P0 ;
8  MOV R33, c[0x0][0x1b0] ;
9  LDGDEPBAR ;
10
```

Listing 4. Triton

```
1  IMAD .WIDE R18, R9, 0x80, R10 ;
2  LDGSTS .E.BYPASS.128 [R219+0x4000], desc[UR16][
     R10.64], P0 ;
3  LDGSTS .E.BYPASS.128 [R219+0x4800], desc[UR16][
     R44.64], P0 ;
4  IMAD .WIDE.U32 R16, R222, 0x2, R64 ;
5  LDGSTS .E.BYPASS.128 [R219+0x5000], desc[UR16][
     R46.64], P0 ;
6  LDGSTS .E.BYPASS.128 [R219+0x5800], desc[UR16][
     R48.64], P0 ;
7  IMAD .WIDE.U32 R10, R222, 0x2, R60 ;
8  MOV R33, c[0x0][0x1b0] ;
9  LDGDEPBAR ;
10
```

Listing 5. SIP

SUMMARY

SIP:

- Automatically optimize GPUs-native instruction schedules

TODOs:

- Apply static analysis to guarantee end-to-end equivalency
- Investigate better search algorithms

Questions? Email:

gh512@cam.ac.uk

Thank you!