Kernel Origami for Accelerated Scientific Computing

Ryan Bass, Dr. Tyler Allen (Advisor)
Department of Computer Science, University of North Carolina at Charlotte

Motivations

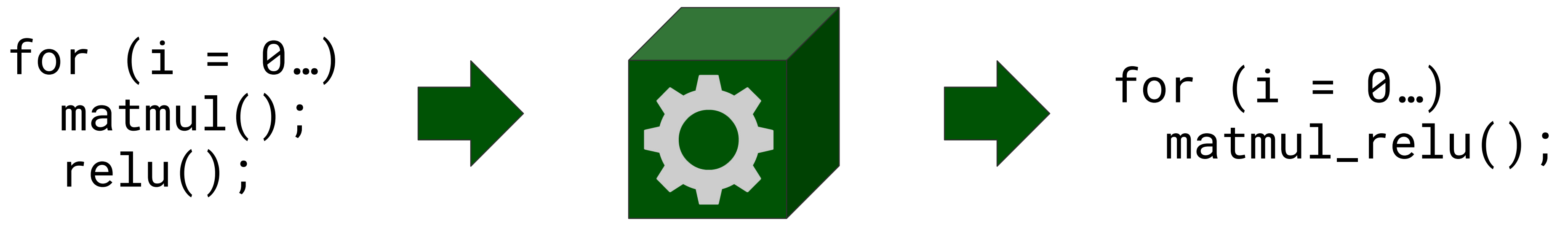
- There is a demand for increased computing power
- Graphics Processing Units (GPUs) are used to gain performance
- Existing software is typically optimized for CPUs, not GPUs
- Performance losses can add up as the problem size increases
- By maximizing performance, different scientific fields could benefit from the increase in computational power

Background

- We propose a method that automatically combines GPU functions
- This would be done in the compiler, the program that converts code into computer instructions
- Combining separate functions can speed up programs
- When combined, there are less memory accesses than when separate
- Less memory accesses means faster runtimes and better performance

Method - Kernel Origami

- As a proof of concept, we are comparing GPU kernel runtimes
- We built two kernels using Cutlass, a GPU library that supports common operations like GEMM (General Matrix Multiplication)
- One GEMM kernel calls the matrix function, then an activation function separately; the other combines the two into one function
- The kernel with the combined functions should be faster than the one with the separate functions, due to less memory accesses

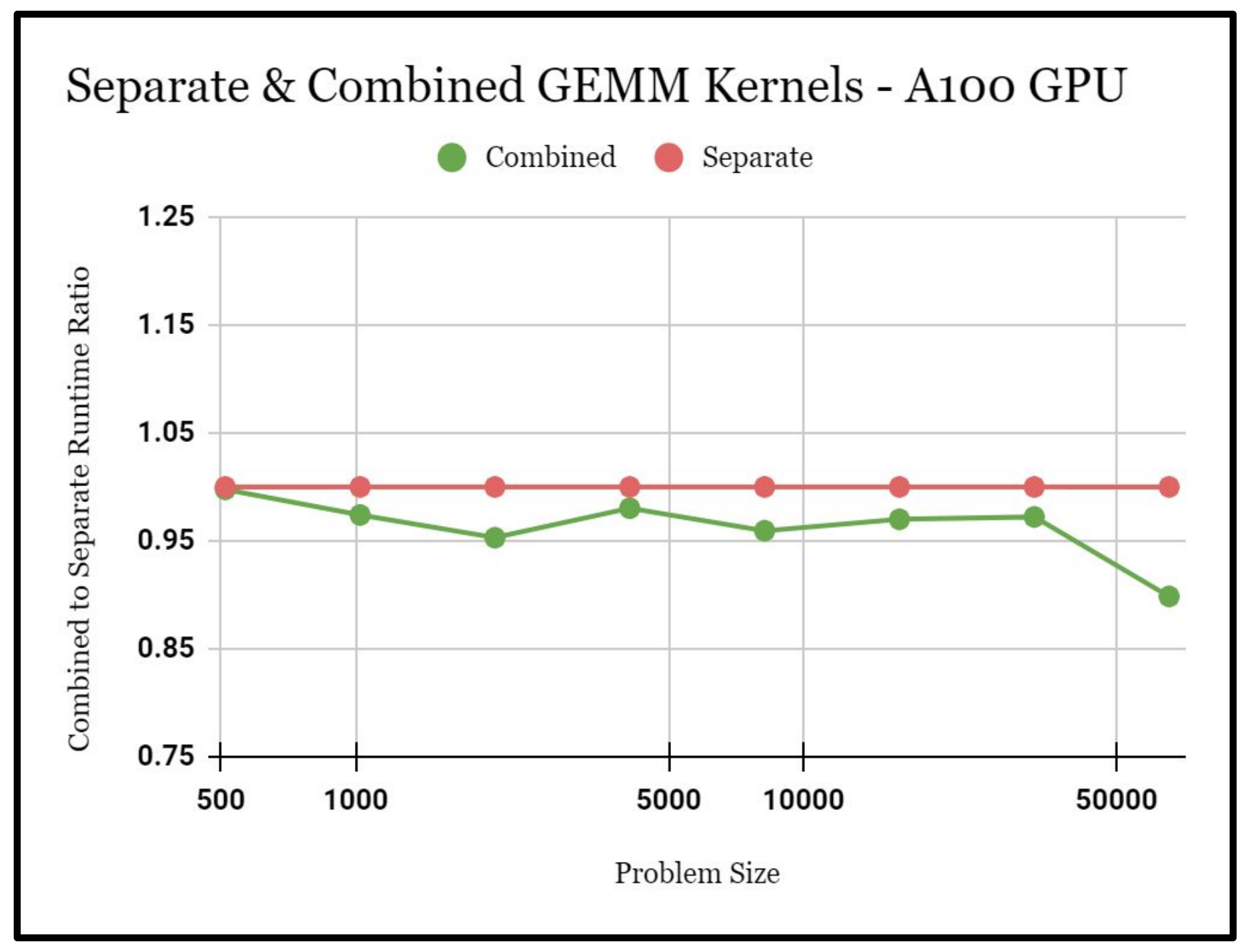


```
for (i = 0…)
  matmul();
  relu();
```

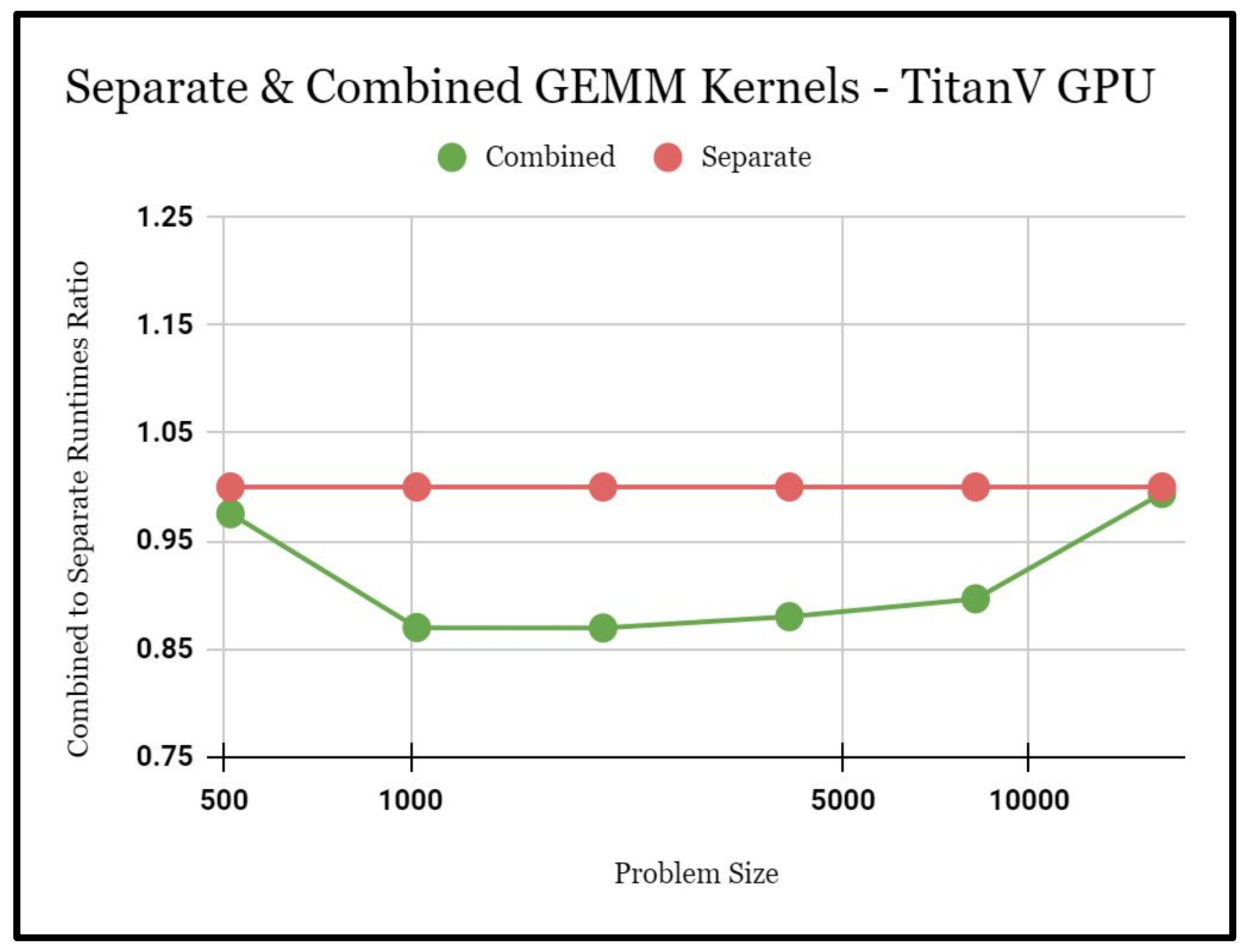
```
for (i = 0…)
  matmul_relu();
```

Results

- Our results show that the combined kernel is faster than the separate kernel, with problem sizes ranging from 512 to 65536
- The combined kernel runs up to 10% faster than the separate kernel



Separate & Combined GEMM Kernels - A100 GPU



Separate & Combined GEMM Kernels - TitanV GPU

Conclusions

- We found that this optimization technique is promising
- If this is implemented at a large scale, it could speed up execution time, providing needed results faster

Kernel	4096	8192	16384
Separate (ms)	13.94	101.45	735.99
Combined (ms)	12.27	90.93	731.57

Future Plans

- Now that the proof of concept has been established, we will move towards creating a compiler solution
- The goal of this compiler is to automatically combine GPU functions
- In order to implement this, we need to determine if two functions can be combined, and then combining them
- Extensive testing and optimization would need to be done to ensure no functionality is lost