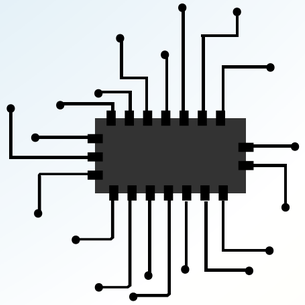
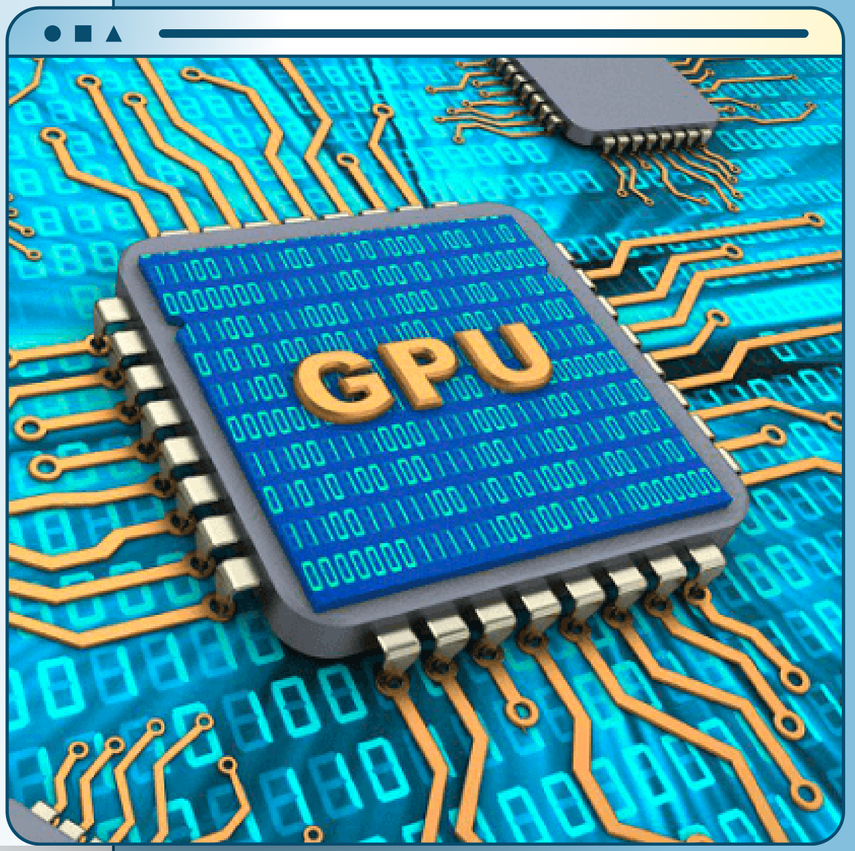
GENERAL PURPOSE COMPUTING ON GRAPHICS PROCESSING UNITS



Presented by: Dhivyan Sureshkumar

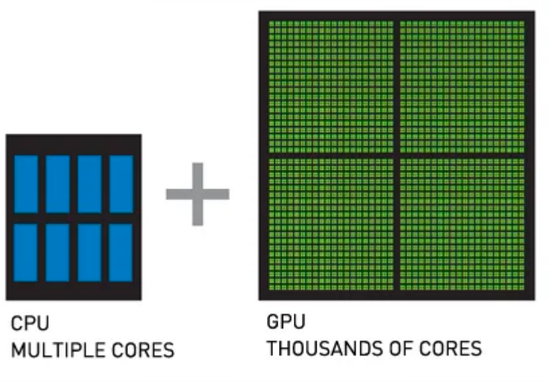
OVERVIEW

- Introduction & Background
- Problem Statement & Hypothesis
- Related work discussion
- Methodology
- Results
- Implementation & demonstration



INTRODUCTION & BACKGROUND

TITLE: Optimising CUDA code by kernel fusion: application on BLAS



- A **GPU (graphics processing unit)** is a computer chip that generates graphics and images by performing fast mathematical computations (Kruglov et al., 2016)
- The purpose of GPGPUs are to grasp the power of GPUs and to carry out tasks previously done by **central processing units (CPUs)**
- **CUDA GPUs** are good at performing arithmetic operations but face challenges with memory access
- Memory-bound kernels spend a lot of time reading from and writing to global memory, leading to performance bottlenecks
- To address this issue, in this study the researchers aim to **automate the fusion of kernels** (through source-to-source compilers) to reduce memory transfers and enhance overall performance of GPU computations (Filipovič et al., 2015)

GPGPU

PROBLEM STATEMENT & HYPOTHESIS

Problem Statement

- The main issue revolves around the performance bottlenecks associated with memory-bound GPU computations
- This occurs due to substantial time it takes to retrieve data from and store data in global memory
- The issue is worsened by the growing gap between GPU's computational abilities and its memory transfer speed in modern GPU systems

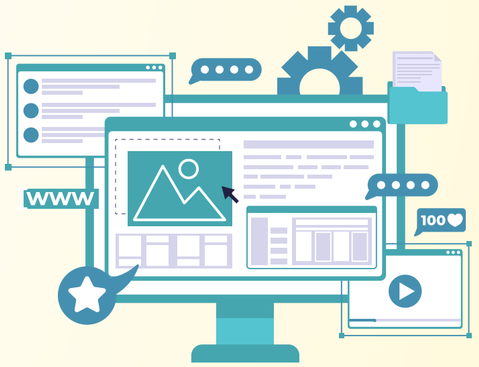
Hypothesis

- It is hypothesised that automated kernel fusion with specific characteristics such as kernels that perform map and reduce operations can increase the efficiency and enhance performance of the GPU memory-bound computations

(Filipovič et al., 2015)

RELATED WORK DISCUSSION

- **The implementation of SkeTo and Trust (Bell & Hoberock, 2011; Sato & Iwasaki, 2009)**



✓ Frameworks that promote GPU kernel fusion

✗ Doesn't allow for larger data to be processed (matrix tiles)

- **The use of GROPHECY and similar performance projection tools (Meng et al., 2011; Wahib & Maruyam, 2014)**

✓ Fuses kernels when it is expected to gain performance

✗ Absence of compiler to automatically generate fused kernels

- **The application of BTO BLAS and DESOLA compilers (Belter et al., 2009; Russell et al., 2011)**

✓ Fuses BLAS functions, thus targeting modern CPUs

✗ Fails to address various computational problems

METHODOLOGY

1. **Sequences selected -** linear algebra kernels from BLAS each with different computations
2. **GPU platforms -** Performance is measured on 3 different GPU architectures - Fermi (Tesla M2090), Kepler (Tesles K20) and Maxwell (GeForce GTX980)
3. Experiments are conducted using CUDA 6.5 and Error-correcting code (ECC) is enabled on the Tesla GPUs
4. The selected sequences in both single/double precision (SP/DP) arithmetic are measured by comparing the performance of hand-tuned CUBLAS implementations with kernels generated by the compiler
5. The study benchmarks the performance of sequences that are equivalent to CUBLAS operations

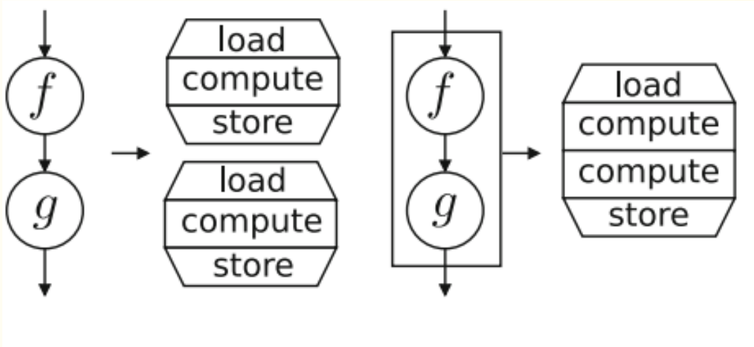


Illustration of a simple fusion

(Filipovič et al., 2015)

ANALYSIS OF RESULTS

- **Significant performance improvement**

Single precision operations - 0.94 times to 2.24 times the speed of CUBLAS

Double precision operations - 0.76 times to 2.04 times

- **Impact of the fusion speedup**

Fusion speedup ranged from 1 to 2.76 times which indicates the effectiveness of kernel fusion in enhancing GPU code performance

- **GPU-generated code VS CPU-based BLAS (BTO-BLAS)**

The GPU-generated code developed in the study achieved a higher speedup compared to BTO BLAS

Overall, the results indicate kernel fusuon as a powerful technique for optimising GPU code

Table 5 Fusion speedup

	Fermi (M2090)		Kepler (K20)		Maxwell (GTX980)	
	SP	DP	SP	DP	SP	DP
AXPYDOT	1.25×	1.28×	1.39×	1.22×	1.28×	1.25×
BiCGK	1.81×	1.81×	1.52×	1.53×	1.95×	1.78×
GEMVER	2.15×	2.35×	2.00×	2.76×	2.67×	2.08×
GESUMMV	1.05×	1.01×	1.01×	1.00×	1.02×	1.01×
VADD	1.47×	1.47×	1.75×	1.53×	1.53×	1.53×
WAXPBY	2.51×	2.33×	2.62×	2.36×	2.36×	2.36×

Table 6 Range of speedups measured with BTO BLAS and our compiler

Sequence	BTO speedup	Our speedup
AXPYDOT	1.22 × –1.58 ×	1.67 × –1.85×
ATAX	1.05 × –1.37 ×	0.77 × –1.11×
BiCGK	1.12 × –1.50 ×	1.18 × –2.02×
GEMV	0.61 × –0.83 ×	0.80 × –1.15×
GEMVT	0.94 × –1.29 ×	0.77 × –1.11×
GEMVER	2.04 × –2.37 ×	1.53 × –2.24×
GESUMMV	0.75 × –0.93 ×	0.76 × –1.33×
MADD	1.35 × –1.47 ×	0.92 × –1.59×
VADD	1.13 × –1.83 ×	1.48 × –2.14×
WAXPBY	1.18 × –1.88 ×	1.93 × –2.05×

(Filipovič et al., 2015)

REFERENCES

- Bell, N., & Hoberock, J. (2011). Thrust : A Productivity-Oriented Library for CUDA 26.
- Belter, G., Jessup, E. R., Karlin, I., & Siek, J. G. (2009). Automating the generation of composed linear algebra kernels. Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. https://doi.org/10.1145/1654059.1654119
- Filipovič, J., Madzin, M., Fousek, J., & Matyska, L. (2015). Optimizing Cuda Code by kernel fusion: Application on blas. The Journal of Supercomputing, 71(10), 3934–3957. https://doi.org/10.1007/s11227-015-1483-z
- Kruglov, A., Chiryshev, A., & Shishko, E. (2016). Applying of the nvidia cuda to the video processing in the task of the Roundwood Volume Estimation. *ITM Web of Conferences, 8*, 01009. https://doi.org/10.1051/itmconf/20160801009
- Meng, J., Morozov, V. A., Kumaran, K., Vishwanath, V., & Uram, T. D. (2011). Grophecy. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. https://doi.org/10.1145/2063384.2063402
- Russell, F. P., Mellor, M. R., Kelly, P. H. J., & Beckmann, O. (2011). DESOLA: An active linear algebra library using delayed evaluation and runtime code generation. Science of Computer Programming, 76(4), 227–242. https://doi.org/10.1016/j.scico.2008.06.002
- Sato, S., & Iwasaki, H. (2009). A skeletal parallel framework with Fusion Optimizer for GPGPU programming. Programming Languages and Systems, 79–94. https://doi.org/10.1007/978-3-642-10672-9_8
- Wahib, M., & Maruyama, N. (2014). Scalable kernel fusion for memory-bound GPU applications. SC14: International Conference for High Performance Computing, Networking, Storage and Analysis. https://doi.org/10.1109/sc.2014.21

IMPLEMENTATION & DEMONSTRATION