PPOPP 2020

Optimizing GPU Programs By Partial Evaluation

Semyon Grigorev

JetBrains Research, Saint Petersburg University, Russia s.v.grigoriev@spbu.ru, Semyon.Grigorev@jetbrains.com



Problem Statement

Memory traffic is a bottleneck of GPGPU programms. There are cases of data analysis when some of kernel parameters are fixed during many kernel runs.

- Patterns in substring matching
- HMM in homology search
- Query in graph database qurying

Known parameters are still increase memory traffic. Can we automatically opimize procedire when parameters are partially known?

Results

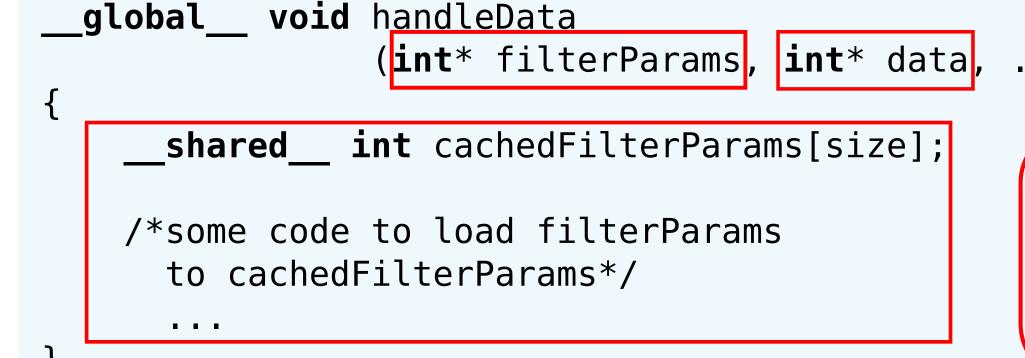
- It is possible to optimize procedures with partially known parameters by using **partial** evaluation [1]
 - Optimized procedure for substring matching is up to 2 times faster
 - _ !!!

Future Research

- Switch to CUDA C partial evaluator.
 - LLVM.mix: partial evaluator for LLVM IR.
- Reduce specialization overhead to make it applicable in run-time.
- Integrete with shared memory register spilling [2].
- Evaluate on real-world examples.
 - Homology search in bioinformatics.
 - Graph processing.

Example

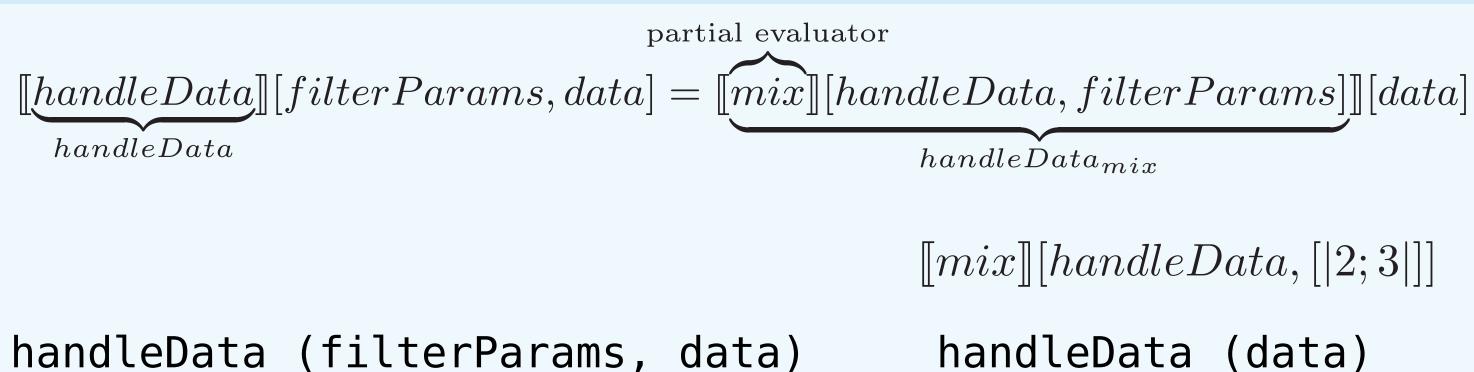
Parameters of filter are fixed during one data processing session which may contains many procedure runs.



In real-world cases we have a huge number of data chunks. Thus we have multiple procedure runs.

Filter params are read only and common for all threads, so we usually copy it into shared memory to reduce memory traffic. In some cases this data can be placed in the constant memory

Partial Evaluation [1]



{
 res = new List()
 for d in data
 for e in filterParams
 if d % 2 == 0 ||
 if d % e == 0
 then res.Add(d)
 return res
}

We Need More Real-World Data

Graph: classical ontologies (RDFs)

Query: same-generation query over type and SubClassOf relations

Grammar: $S \to scor S sco \mid tr S t \mid scor sco \mid tr t$

RDF			Algorithms						
Name	#V	$\#\mathrm{E}$	Scipy	M4RI	GPU	CuSprs	CYK		
atm-prim	291	685	3 ms	2 ms	1 ms	269 ms	8.5 min		
biomed	341	711	$3 \mathrm{ms}$	$5~\mathrm{ms}$	$1 \mathrm{ms}$	$283 \mathrm{ms}$	7.1 min		
pizza	671	2604	6 ms	$8 \mathrm{ms}$	$1 \mathrm{ms}$	292 ms	$54 \min$		
wine	733	2450	$7~\mathrm{ms}$	6 ms	1 ms	294 ms	68 min		

- 2019 (GPU) is 10⁶ times faster than 2016 (CYK) on real-world data
 - Reasonable time even for CPU based implementations
- We should find bigger RDFs
- We should find other real-world cases for CFPQ
 - Both graphs and queries

We Should Do More Research on the Algorithms Scaling

	Graph	Scipy	M4RI	GPU	CuSprs
C	G10k-0.001	37 s	2 s	0.2 s	35 s
Sparse graphs are generated by GTgraph	G10k-0.1	601 s	1 s	$0.1 \mathrm{s}$	$395 \mathrm{s}$
Query: $S \rightarrow a \ S \ b \mid a \ b$	G40k-0.001	_	97 s	8.1 s	_
Query. $D \rightarrow a D b \mid a b$	G80k-0.001	_	1142 s	65 s	-
Craph is a svala	G25k	_	33 s	5 s	_
Graph is a cycle	G50k	_	$360 \mathrm{\ s}$	44 s	_
Query: $S \to S S \mid a$	G80k	_	$1292 \mathrm{\ s}$	190 s	_

- We can handle graphs with 80k vertices in a reasonable time by using GPGPU
 - Technical bound: GPGPU RAM does not fit bigger graphs
- We should evaluate multi-GPU systems
- We should evaluate distributed solutions
- We should implement a sparse boolean matrices library for GPGPU

Contact Us

Our team:

- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Nikita Mishin: mishinnikitam@gmail.com
- Iaroslav Sokolov: sokolov.yas@gmail.com



Both dataset and implementations are available on GitHub:

References

- [1] Neil D. Jones, Carsten K. Gomard, and Peter Sestoft. Partial Evaluation and Automatic Program Generation. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
- [2] Putt Sakdhnagool, Amit Sabne, and Rudolf Eigenmann. Regdem: Increasing GPU performance via shared memory register spilling. CoRR, abs/1907.02894, 2019.
- [3] Roland Leissa, Klaas Boesche, Sebastian Hack, Arsène Pérard-Gayot, Richard Membarth, Philipp Slusallek, André Müller, and Bertil Schmidt. Anydsl: A partial evaluation framework for programming high-performance libraries. *Proc. ACM Program. Lang.*, 2(OOPSLA):119:1–119:30, October 2018.

Acknowledgments

The research is supported by the Jet Brains Research grant and the Russian Science Foundation grant 18-11-00100

https://github.com/SokolovYaroslav/CFPQ-on-GPGPU