PDTgcomp: Compilation Framework for Data Transformation Kernels on GPU

NC STATE UNIVERSITY
Department of Electrical and Computer Engineering
Tri Nguyen and Michela Becchi (tmnguye7, mbecchi @ncsu.edu)

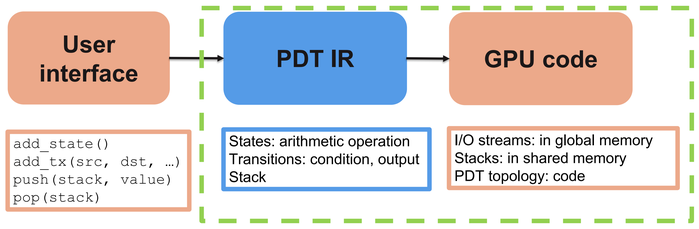
Motivation

- Data transformation kernels are at the core of many data analytics, data processing and scientific applications
- Custom CPU and GPU libraries and hardware accelerators for specific data transformation tasks provide efficiency but lack generality and extensibility
- Accelerating the computational abstraction at the core of data transformation can benefit a broad spectrum of applications

pandas Parquet GNU Scientific Library NVIDIA CUDA RAPIDS

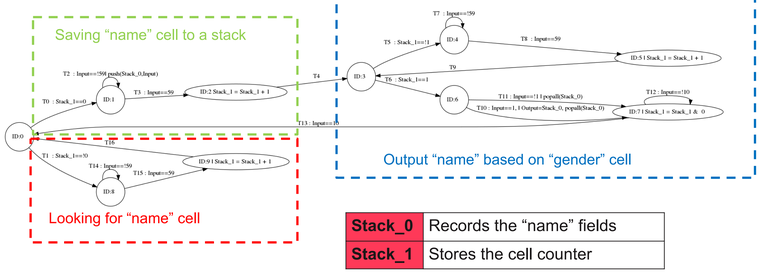
Objective

- Compilation framework that generates accelerated GPU kernels for data transformation tasks expressed using Pushdown Transducers (PDTs)



Pushdown Transducers (PDT)

Example (CSV parsing): PDT that extracts the name of all female members from a CSV file containing "name, gender, occupation" tuples



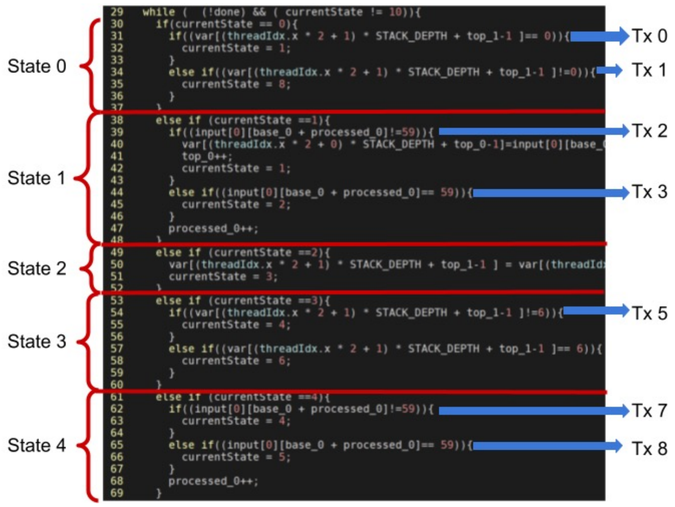
Stack_0	Records the "name" fields
Stack_1	Stores the cell counter

Conversion Steps

1. Convert the list of states into an *if-else* block in which each if-condition guards the content of a state
2. For each state:
   a. convert the associated arithmetic operation (if any) to the appropriate arithmetic operation on stack values
   b. convert the list of outgoing transitions into an *if-else* block, with an if-statement per transition. Generate if-conditions and statements according to the appropriate input, output and stack read/write operation
3. Allocate stacks in shared memory
4. Store the remaining context information (pointers in I/O streams, active state indicator) into local variables to be stored in the register file

Basic Code

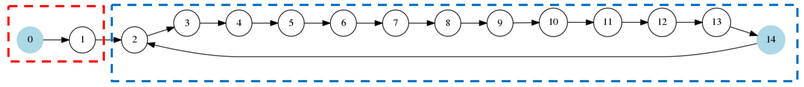
Snapshot of generated PDT traversal code



Optimizations

Goals:
- Reduce code size
- Reduce control flow operations
- Improve memory access patterns

No optimizations

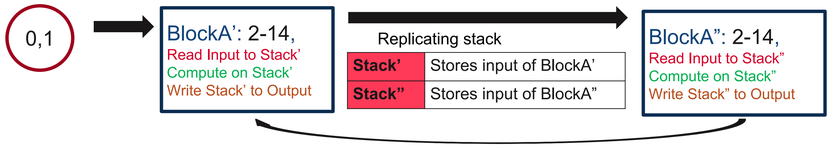


State Merging

- Advantage: control flow reduction
- Disadvantage: not always applicable

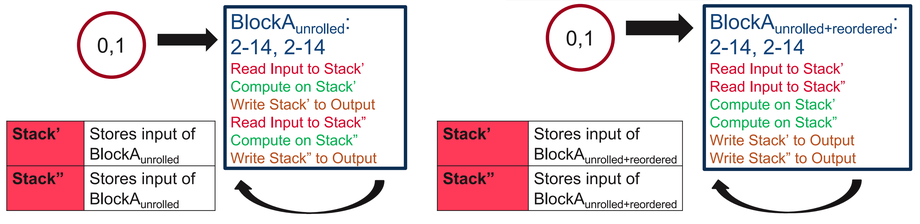
Stack	Stores input of BlockA

Loop Unrolling



Stack'	Stores input of BlockA'
Stack"	Stores input of BlockA"

Instructions Reordering



Stack'	Stores input of $BlockA_{unrolled}$
Stack"	Stores input of $BlockA_{unrolled}$

Stack'	Stores input of $BlockA_{unrolled+reordered}$
Stack"	Stores input of $BlockA_{unrolled+reordered}$

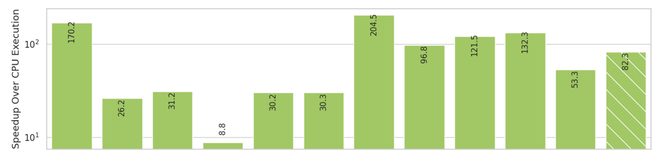
- Advantages: Loop unrolling enables more aggressive optimizations such as reordering of reads (from input) and writes (to output)
- Disadvantages: Loop unrolling increases register pressure

Experimental Setup

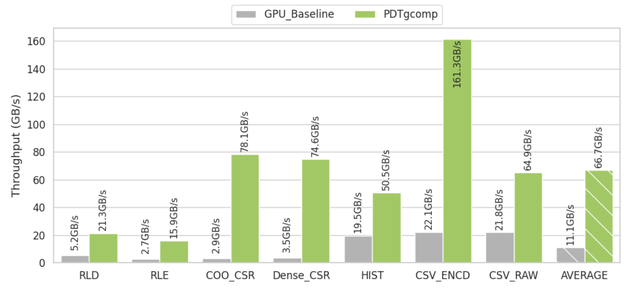
Transformation class	Input dataset	CPU Baseline	GPU Baseline
Data Enc/Dec	Cantebery Corpus	Parquet [1]	Nvidia Thrust [4]
Matrix Transformation	Texas A&M Sparse Matrix	Intel MKL [3]	Nvidia cuSparse [4]
Histogram	RDU Accident and Crime Report	GSL Histogram [2]	Nvidia Cub [4]
CSV Query	RDU Accident and Crime Report	Pandas [5]	Rapids AI [6]

System	
CPU	2x Intel Xeon E5-2630 2.2GHz
GPU	NVIDIA TITAN XP 12GB, 30SMs
OS/CUDA	Ubuntu 18.04, CUDA toolkit 11.7

Performance



PDTgcomp speedup over custom CPU libraries



Throughput of PDTgcomp and custom GPU libraries in GB/s

- Average 82x and 6x speedup over custom CPU and GPU implementations, respectively
- On CPU, highest speedup on matrix transformation (200x) and CSV querying (120x)
- On GPU, highest throughput on CSV querying (161GB/s)

Conclusion and Future Work

- We demonstrated a method to generate efficient GPU code implementing data transformation tasks expressed using PDT
- Future work encompasses more compiler optimizations to further improve performance.

References and Acknowledgement

[1] Apache Parquet: https://parquet.apache.org
[2] GNU scientific library: https://www.gnu.org
[3] Intel MKL: https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-mkl-for-dpcpp/top.html
[4] Cuda toolkit: https://docs.nvidia.com/cuda
[5] Pandas: https://pandas.pydata.org
[6] Open GPU data science: https://rapids.ai
This work was supported by National Science Foundation awards CNS-1812727 and CCF-1907863.