Custom GPU Parser vs cuDF CSV Parser and cuDF Parquet vs CPU Methods

Tarun Sreepada ID: m5281045

University of Aizu

January 16, 2025

Outline

Introduction

- **CSV Files:** A file format for storing tabular data in plain text. Encodings can vary, and line sizes can be inconsistent.

Listing: CSV file

```
1 ,2 ,3 ,4 ,5
6 ,7 ,8 ,9 ,10
11 ,12 ,13 ,14 ,15
```

- **Encoding Table:** A mapping of bits to characters, such as ASCII or UTF-8.

ASCII Code Chart

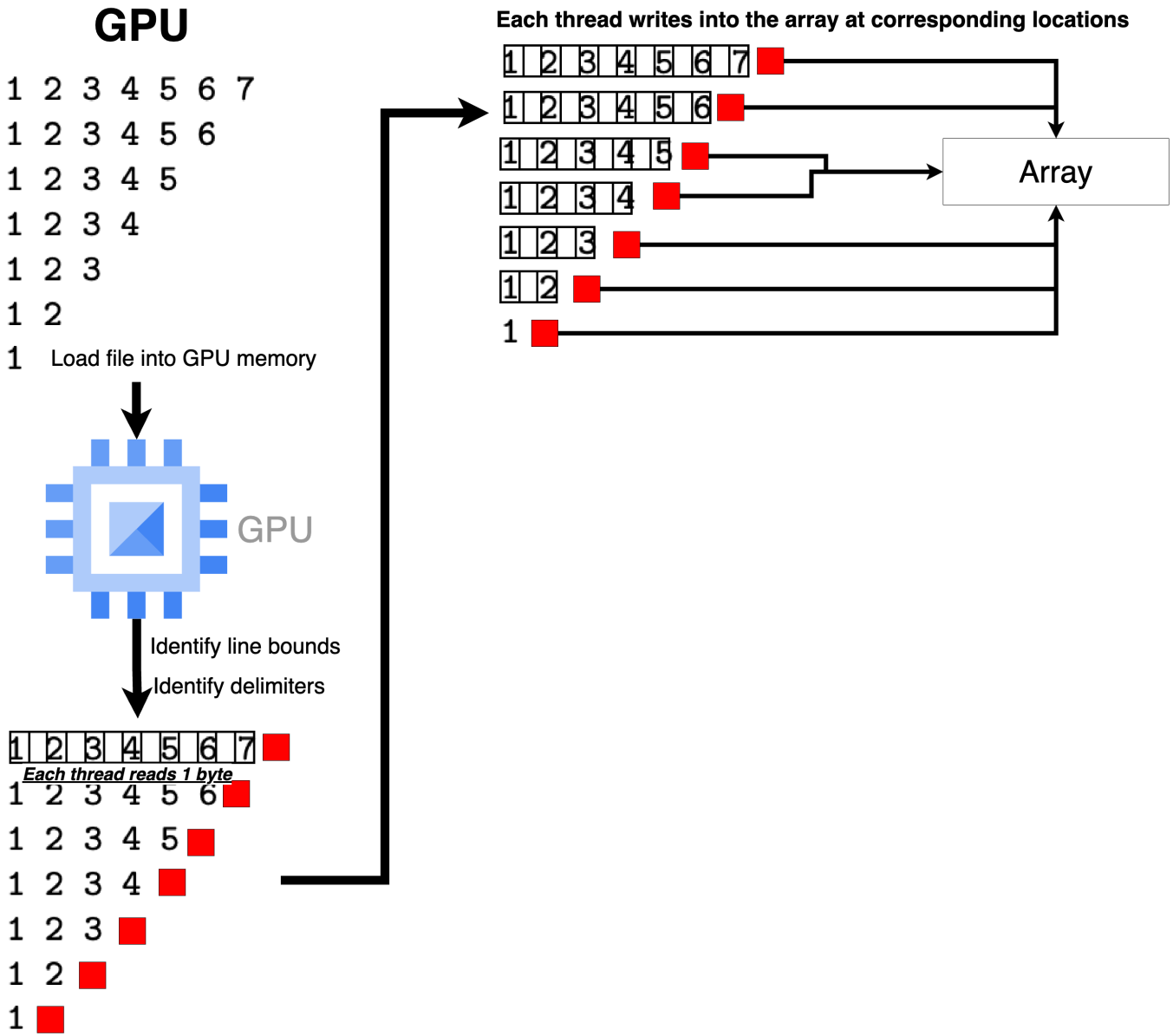
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Figure: Encoding Table

Custom GPU Parser Approach

- **File Loading:**
  - Load the entire CSV file into memory.
- **Transaction Counting:**
  - Count the number of newline characters to determine the total number of transactions.
- **Line Boundary Detection:**
  - Identify the exact positions of all newline characters to know where each record ends.
- **Item Counting and Memory Allocation:**
  - For each newline, count the number of CSV items.
  - Create a new memory area dedicated for the conversion of UTF-8 encoded integers to unsigned integers.
- **Parallel Parsing:**
  - Launch one GPU thread per CSV line (transaction) to parse and convert the data into the newly allocated area.

GPU Read Illustrated



1. Determine number of line bounds and delimiters.

CPU Parser Approach

- **Chunk Splitting:**
  - The file is divided into equal-sized chunks.
- **Boundary Adjustment:**
  - Adjust the starting and ending points of each chunk to ensure that no chunk contains partial (incomplete) lines.
- **Parallel Execution:**
  - The number of chunks is determined by the number of available CPU cores.
  - Each core processes one complete chunk concurrently, parsing the full lines in that segment.

CPU Read Illustrated

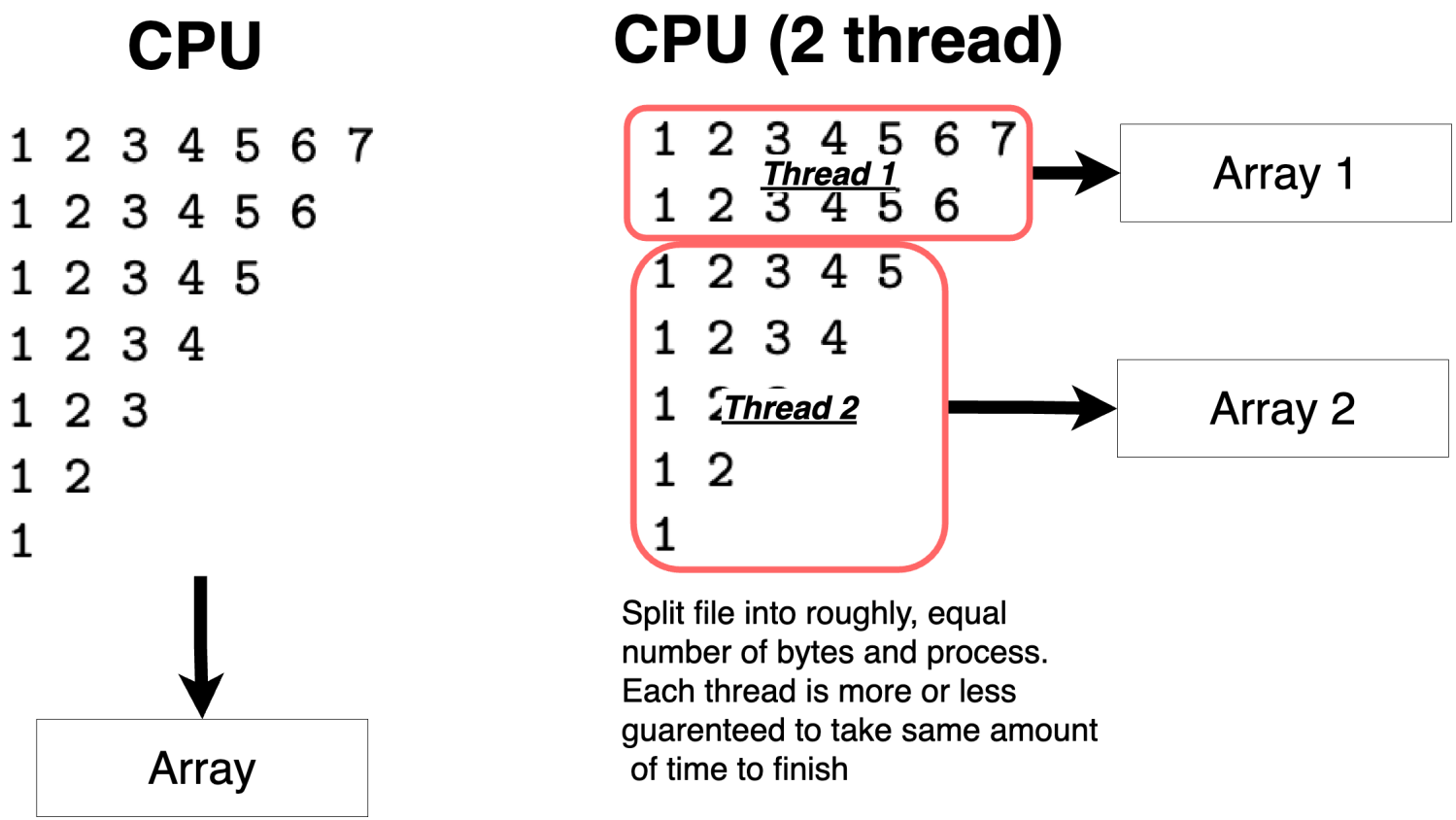


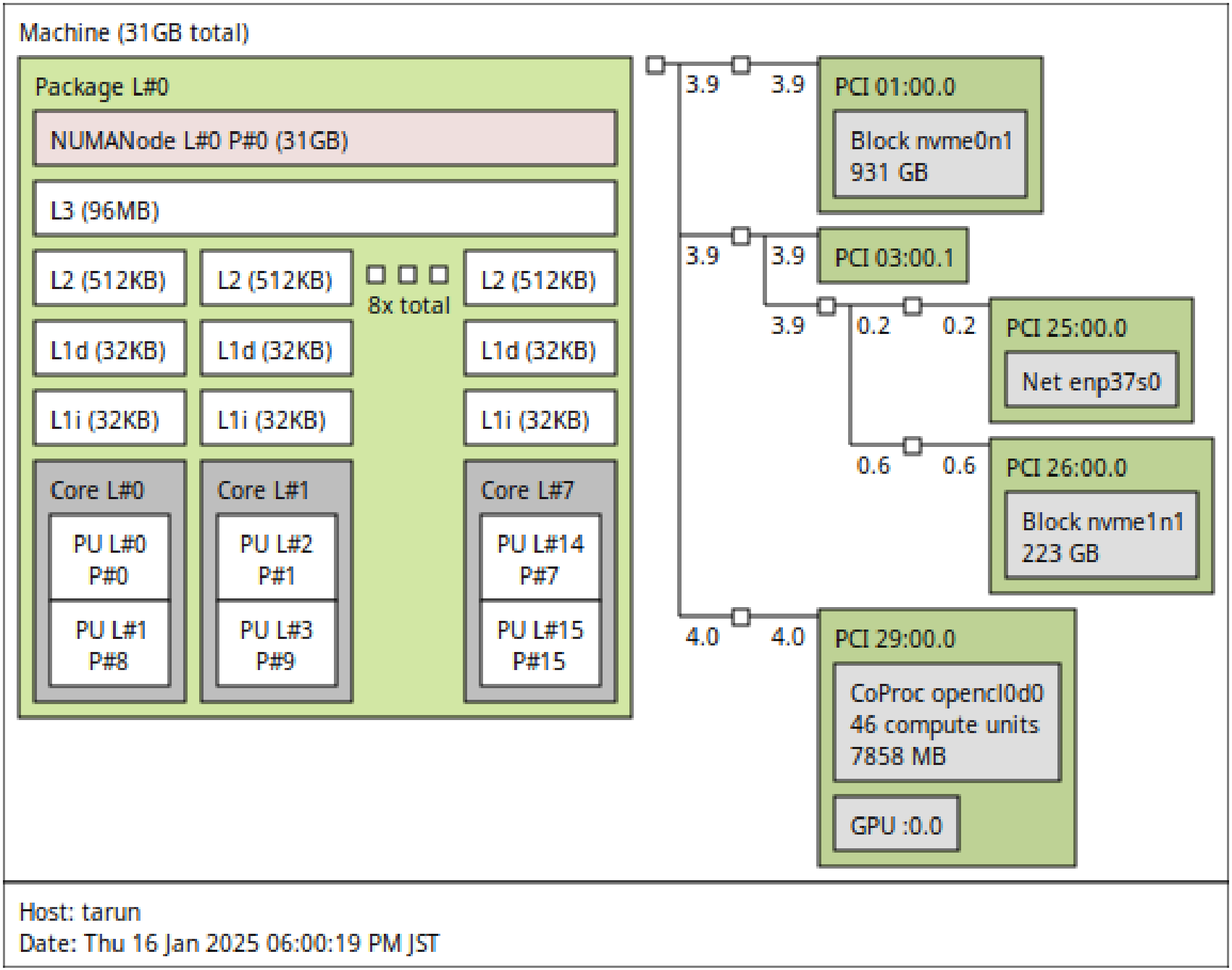
Figure: CPU Read Process

cuDF CSV Parser

- **Integration:** Part of the RAPIDS ecosystem, designed to work seamlessly with other GPU-accelerated libraries.
- **Performance:** Leverages GPU cores to achieve significant speed-ups compared to many CPU-based CSV parsers.
- **Ease of Use:** Simple integration with Python and familiar DataFrame interfaces.

Experimental Setup

- **Datasets:** Synthetic datasets with varying line sizes.
- **Hardware:** AMD Ryzen 5700X3D CPU, NVIDIA RTX 3070 GPU, 32GB RAM, and 256GB SSD (Max Read 400MB/s)
- **Software:** cuDF, cuPy, CPU and GPU parser implementations.
- **Methodology:**
  - Load the datasets using custom GPU parser, cuDF CSV parser, and CPU methods.
  - Measure execution time and resource utilization.


Machine (31GB total)
Package L#0
NUMANode L#0 P#0 (31GB)
L3 (96MB)
L2 (512KB)
L2 (512KB)
8x total
L2 (512KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
Core L#0
PU L#0 P#0
PU L#1 P#8
Core L#1
PU L#2 P#1
PU L#3 P#9
Core L#7
PU L#14 P#7
PU L#15 P#15
3.9
3.9
PCI 01:00.0
Block nvme0n1
931 GB
3.9
3.9
PCI 03:00.1
3.9
0.2
0.2
PCI 25:00.0
Net enp37s0
0.6
0.6
PCI 26:00.0
Block nvme1n1
223 GB
4.0
4.0
PCI 29:00.0
CoProc opencl0d0
46 compute units
7858 MB
GPU :0.0
Host: tarun
Date: Thu 16 Jan 2025 06:00:19 PM JST

- **Triangle:** Files with a variable number of columns per line, emulating datasets with irregular structures.
- **Square:** Files with a uniform number of columns per line, representing regular datasets.

```
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

(a) Example of a Triangle File.

```
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
```

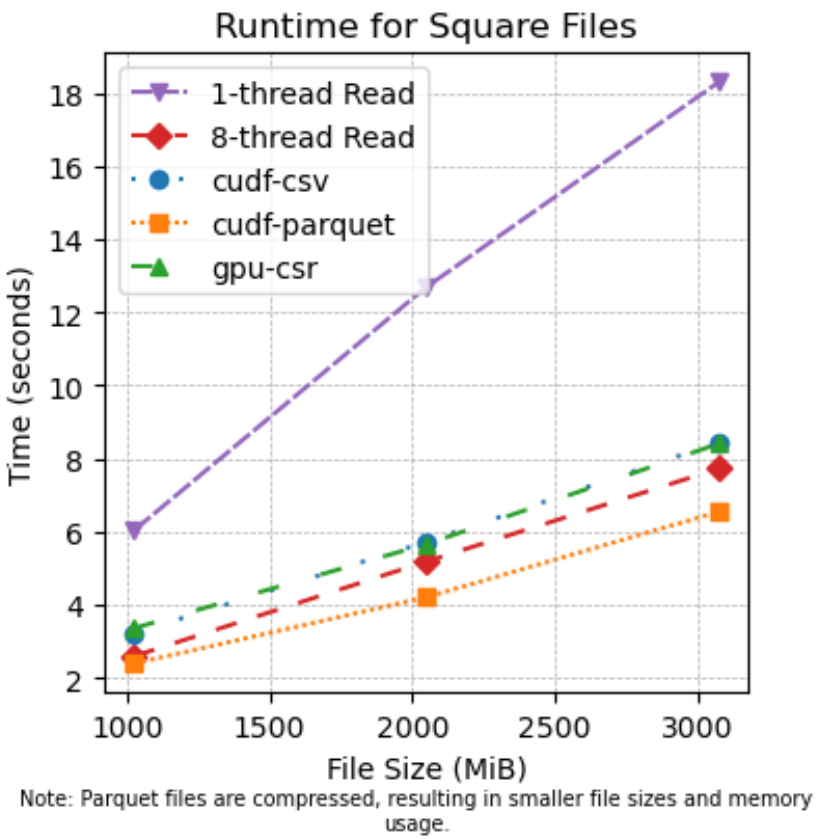
(b) Example of a Square File.

Fig. 1: Examples of Triangle and Square File Structures.

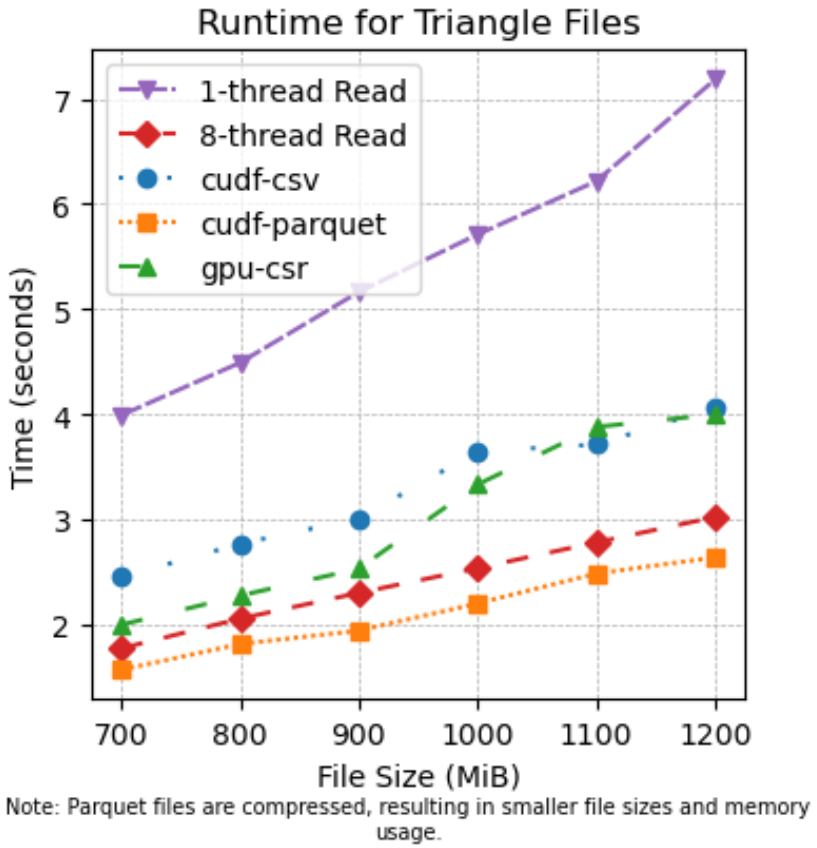
Experimental Results

- **Benchmark Setup:** Parsing large-scale datasets using both methods.
- **Metrics:**
  - Execution Time
  - Resource utilization (GPU vs CPU).
- **Preliminary Findings:**
  - Both the custom GPU parser and cuDF CSV parser show similar speeds as CPU-based parser. SSD is the limiting factor.
  - The custom GPU parser can be more finely optimized for specific data formats (e.g., non-integers) using asynchronous I/O and buffered reads for reduced memory usage.

Experimental Results



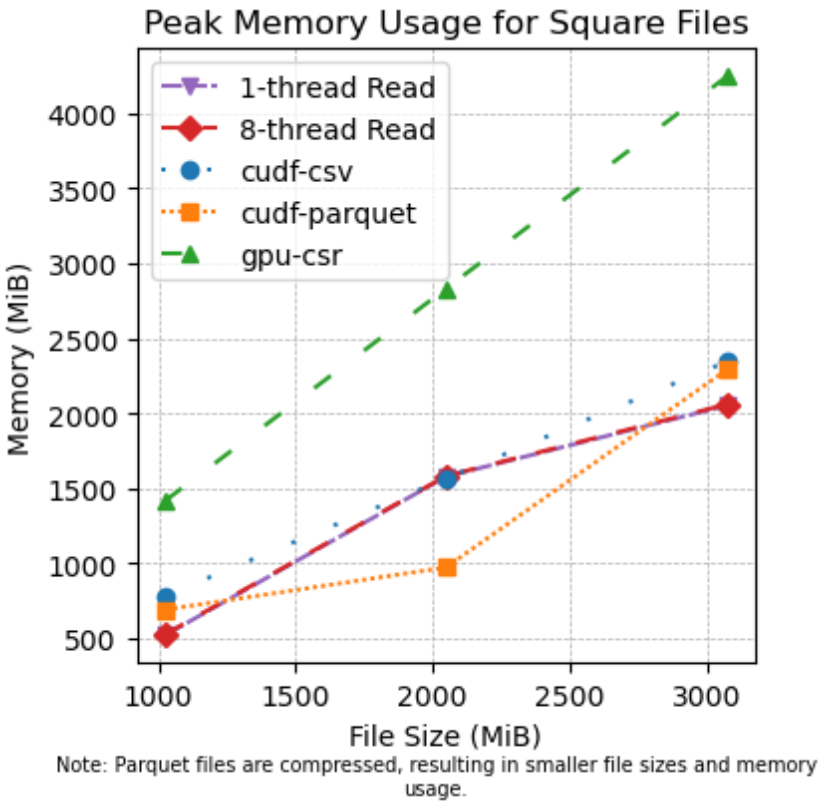
(a) Execution Time: Square Data



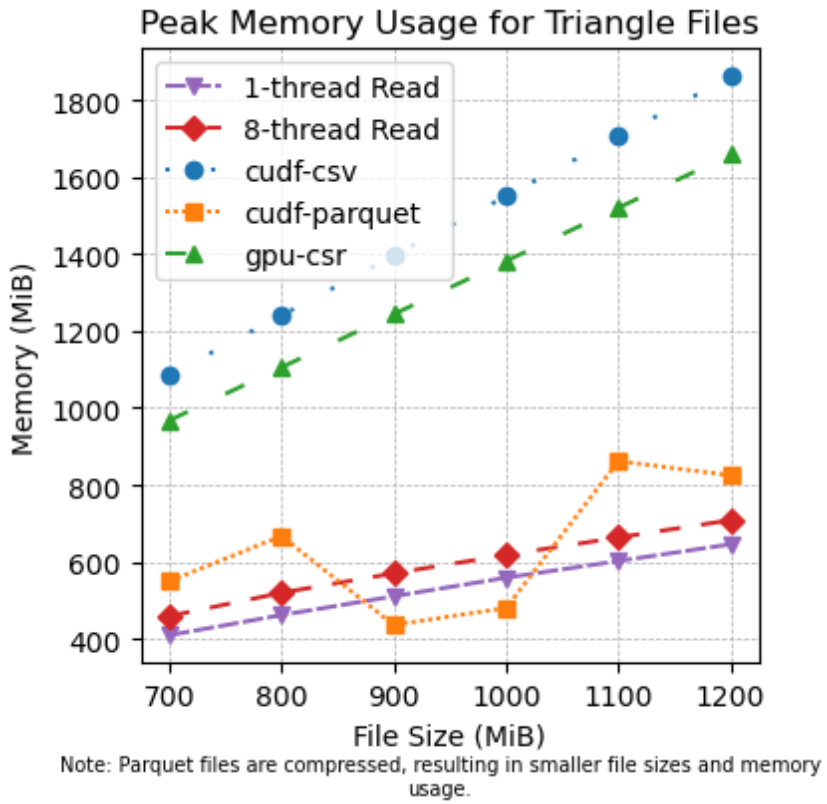
(b) Execution Time: Triangle Data

Figure: Benchmark Comparison of Execution Time and Memory Usage

Experimental Results



(a) Memory Usage: Square Data



(b) Memory Usage: Triangle Data

Figure: Benchmark Comparison of Execution Time and Memory Usage