Universität Bamberg

Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

Universität Bamberg

Faster than before…

Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

Universität Bamberg

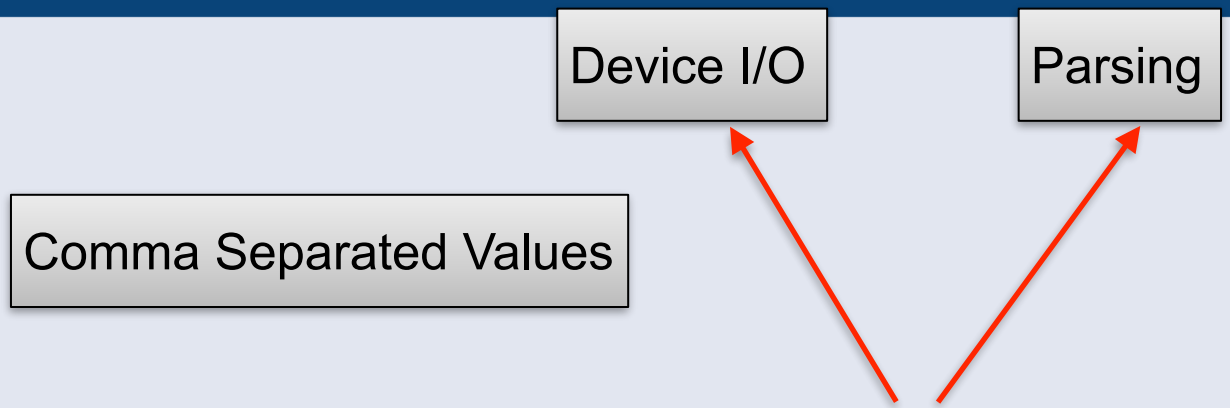
Comma Separated Values

Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

```
ID,Country,ISOCode,Population\n
1,Germany,DE,83149319\n
2,Spain,ES,47007367\n
3,France,FR,67076431\n
4,Italy,IT,60317116\n
5,Japan,JP,126150745\n
6,China,CN,1427647786\n
7,United States of America,US,328239523\n
8,Canada,CA,37894799\n
...
```

Row-based text-file



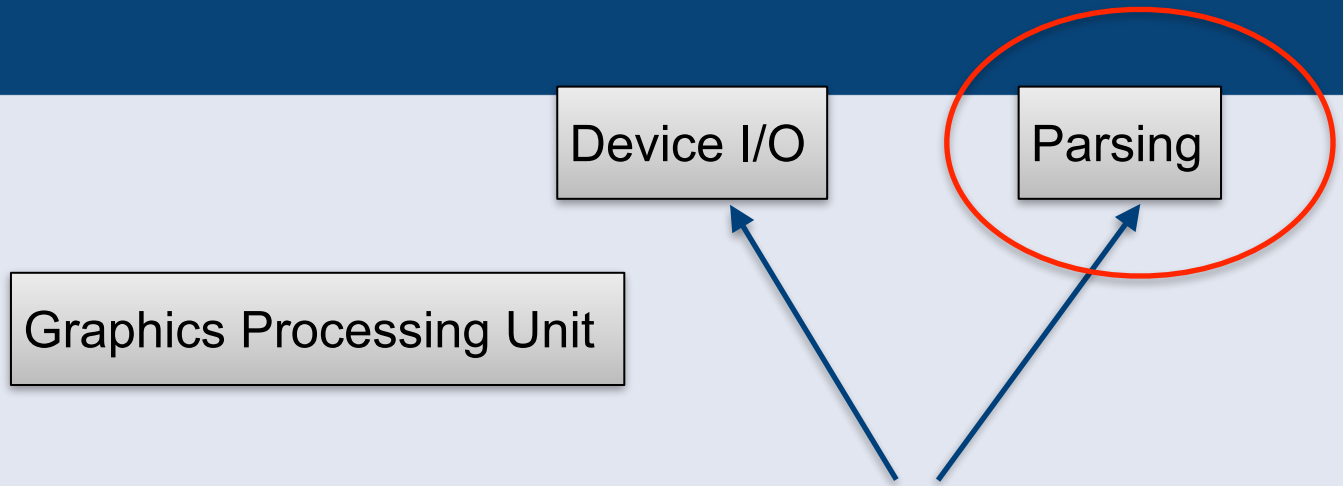
Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

```
ID,Country,ISOCode,Population\n
1,Germany,DE,83149319\n
2,Spain,ES,47007367\n
3,France,FR,67076431\n
4,Italy,IT,60317116\n
5,Japan,JP,126150745\n
6,China,CN,1427647786\n
7,United States of America,US,328239523\n
8,Canada,CA,37894799\n
...
```

Row-based text-file

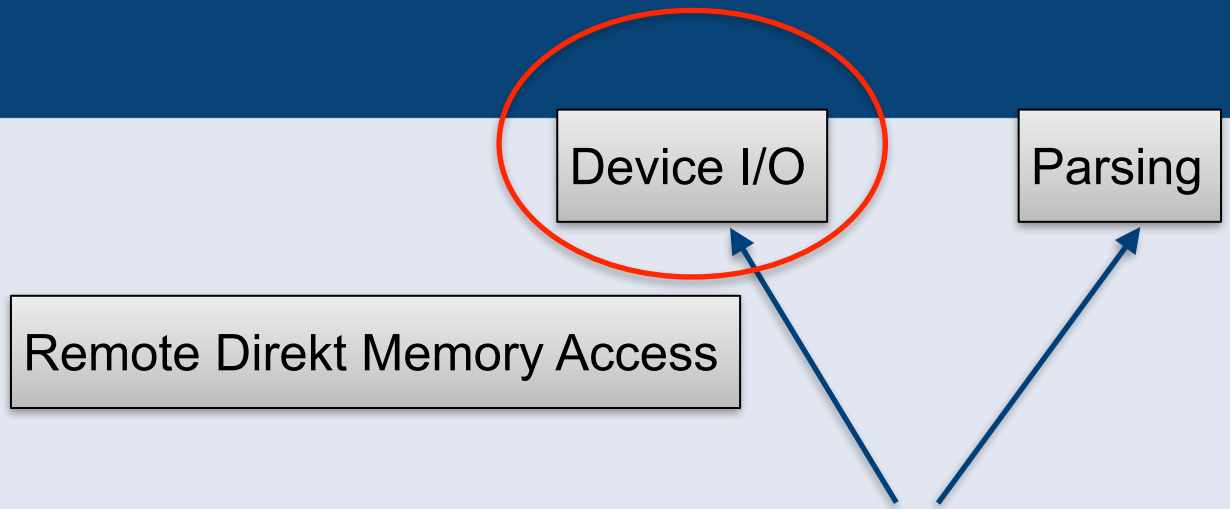
Universität Bamberg



Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

Universität Bamberg



Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

Universität Bamberg

Scope: End-to-End

Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

Wait: Why CSV?

Fast CSV loading using GPUs and RDMA for in-memory data processing

Andreas Haderlein

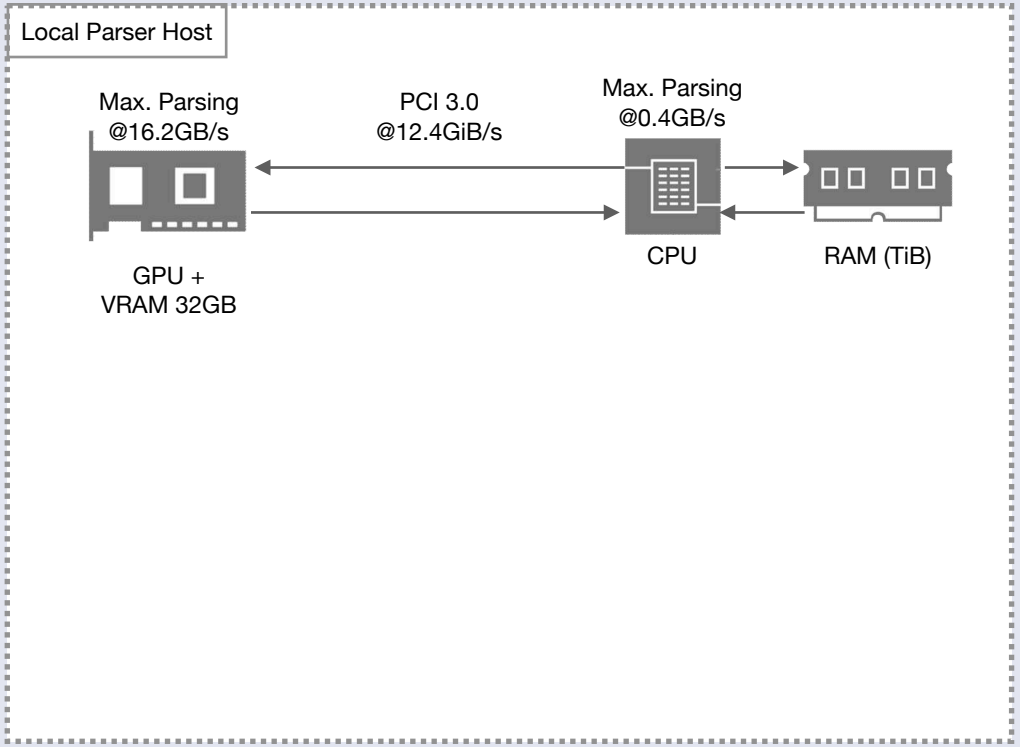
More efficient file formats exist (e.g. Apache Parquet)… So why CSV?

- In use for over 50 years
- platform-independent, other formats lack universal compatibility
- Human-readable and simple

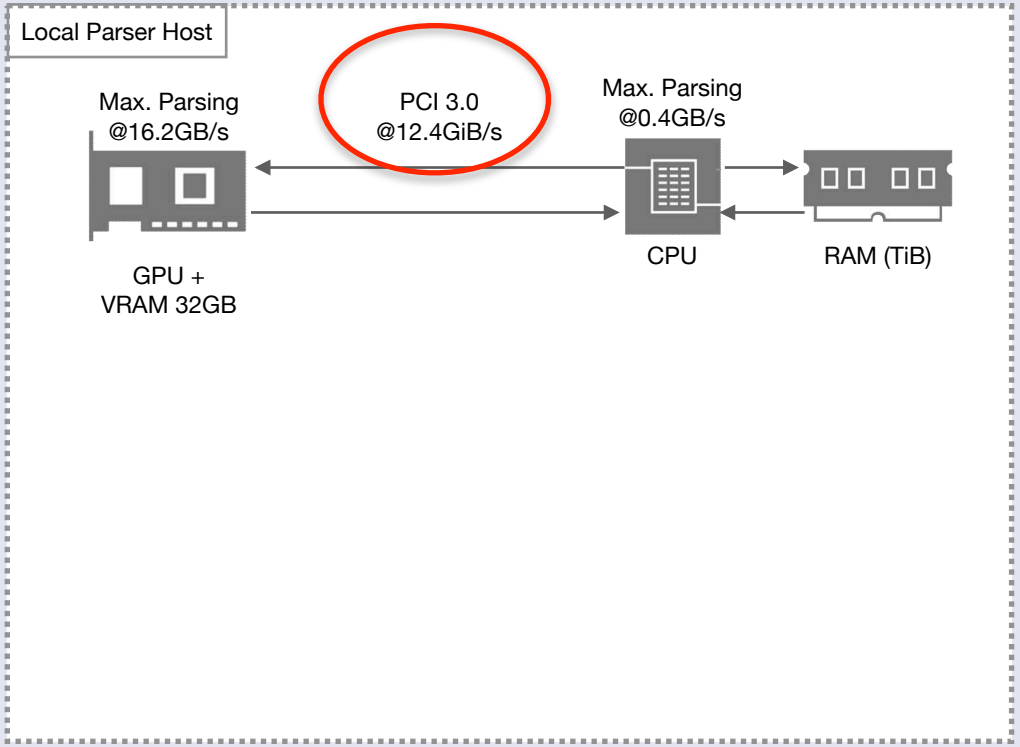
Generally accepted standard does not exist, but RFC 4180 is mostly used.

Table of Content

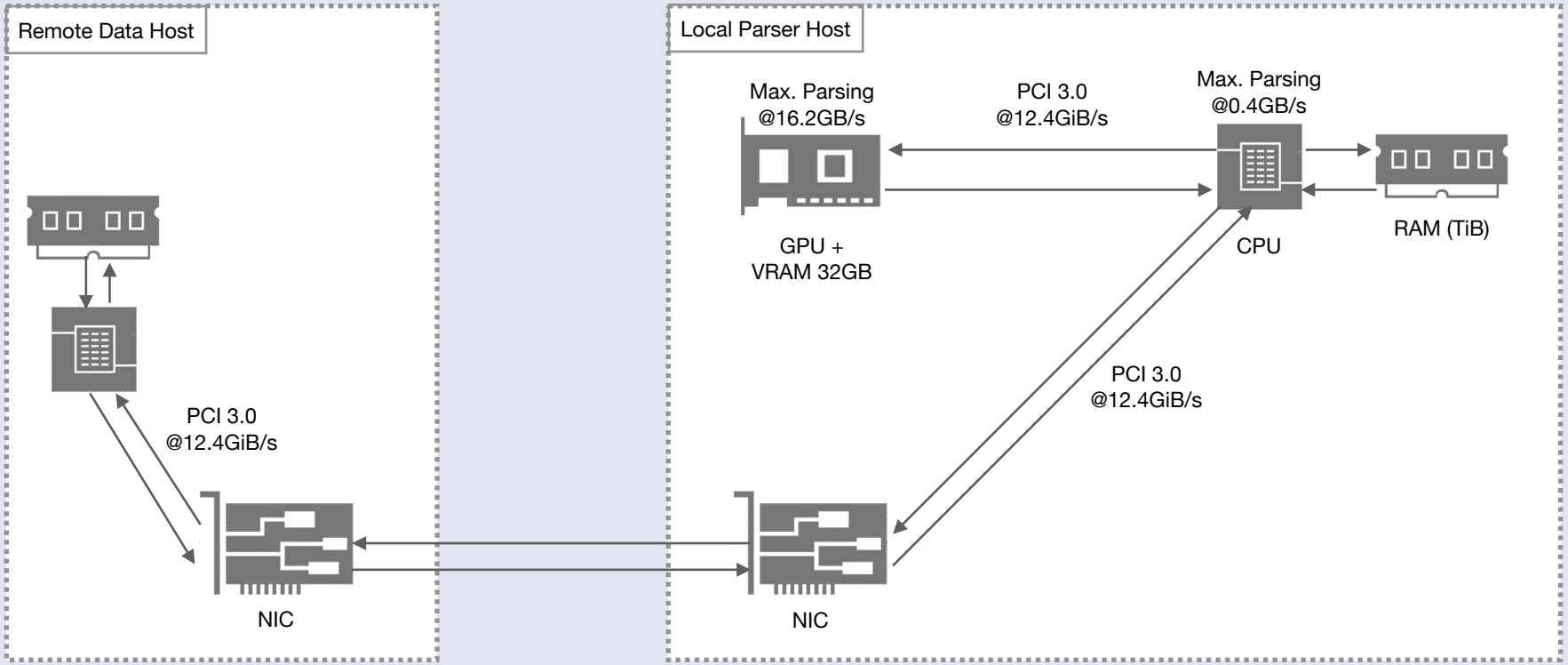
Slow version of reading from main memory



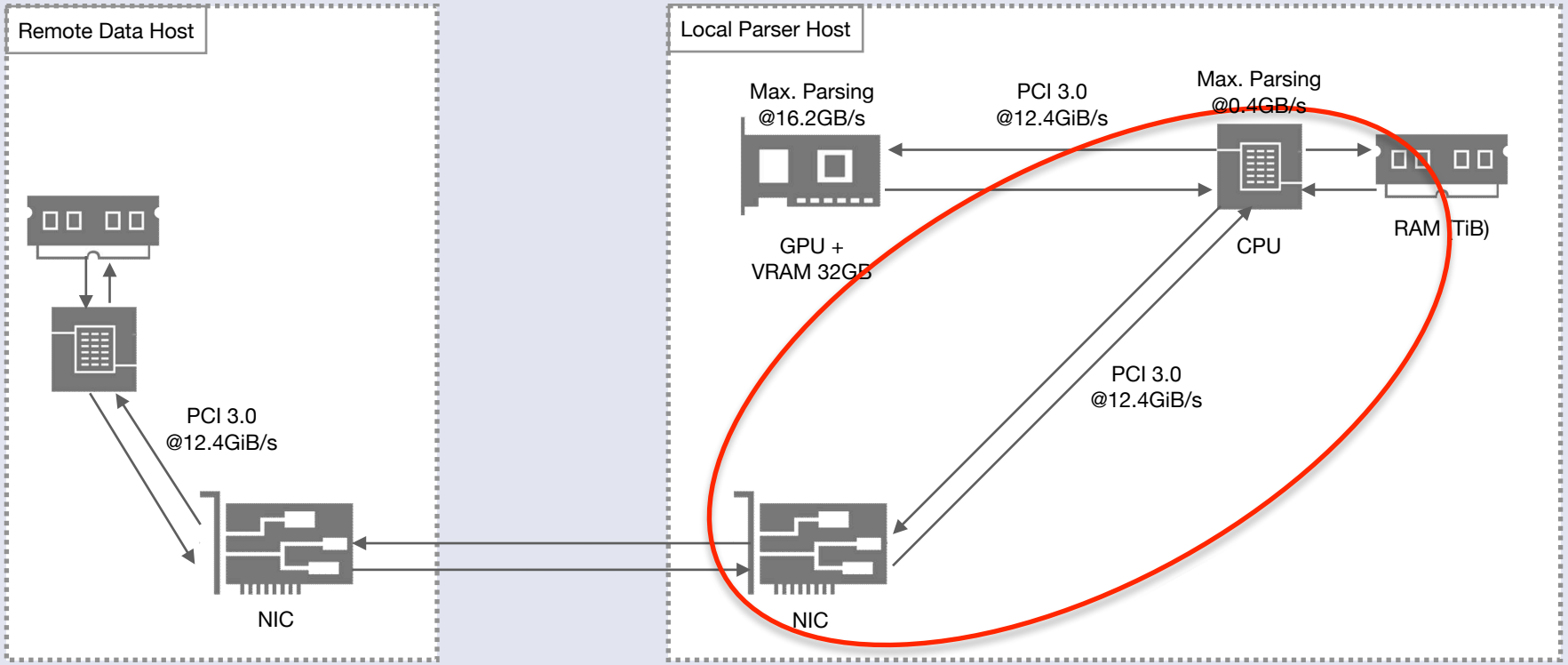
Data transfer bottleneck



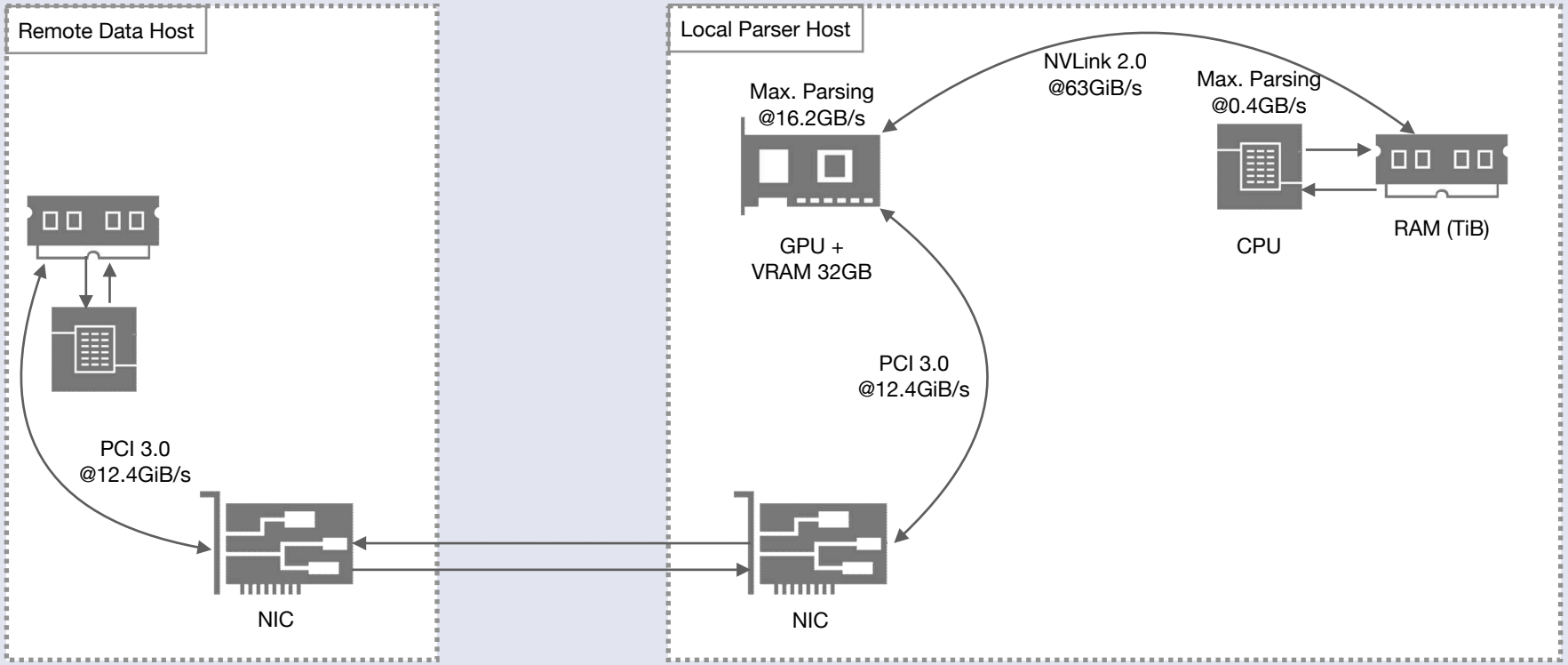
Standard version of receiving data from a data host (Streaming)



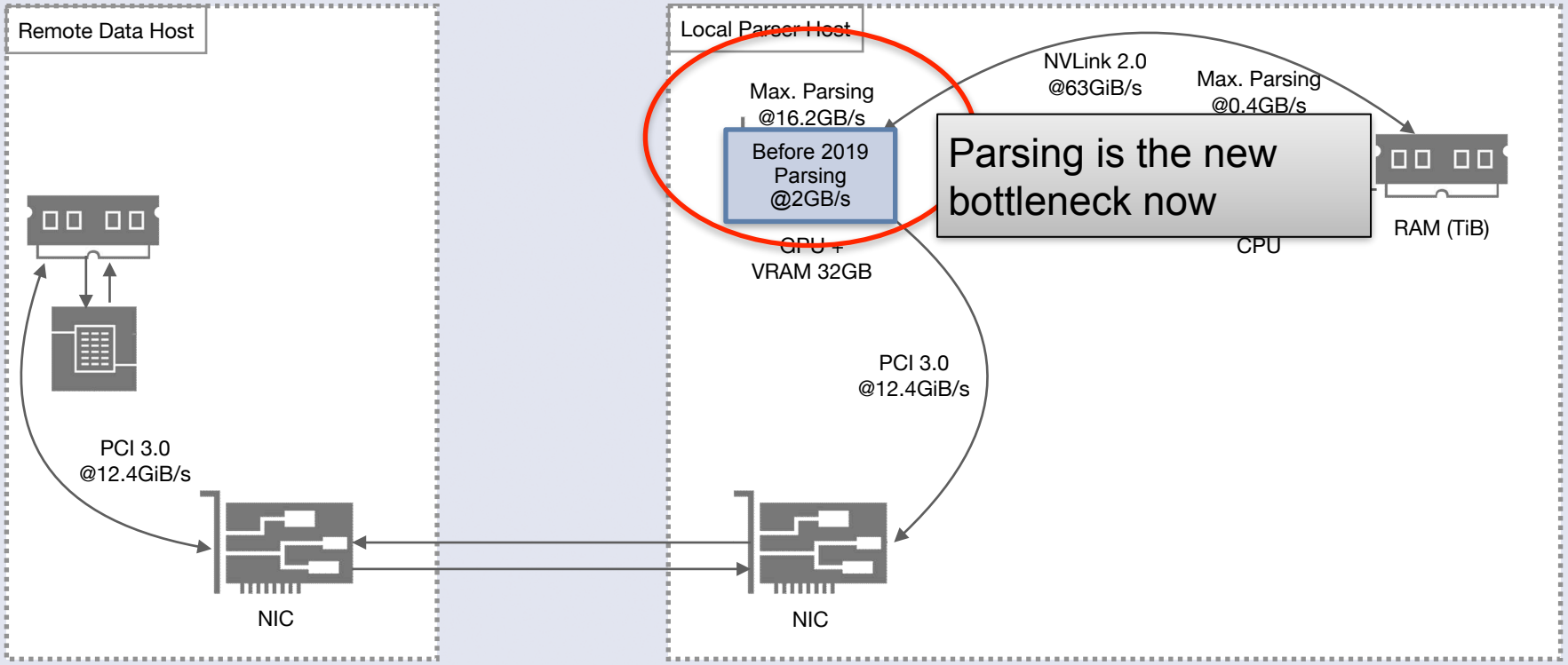
The next bottleneck



Now: GPU interconnect via NVLink 2.0 and GPUDirect via RDMA

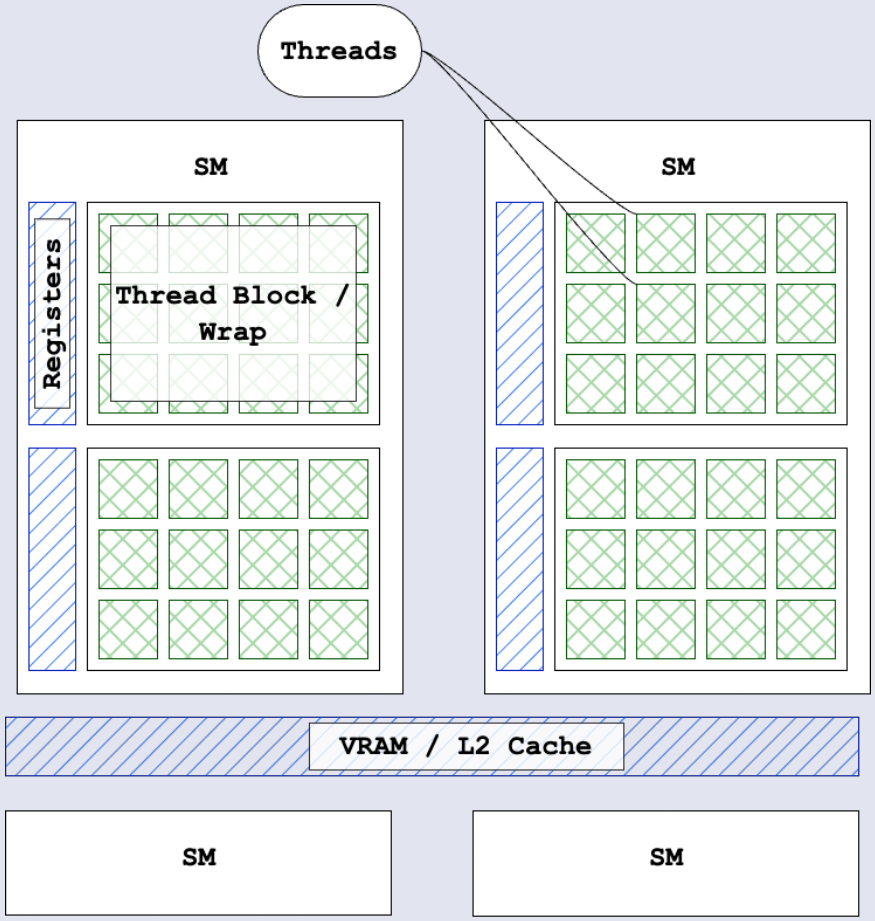


The Problem to be solved in this paper: data loading bottleneck

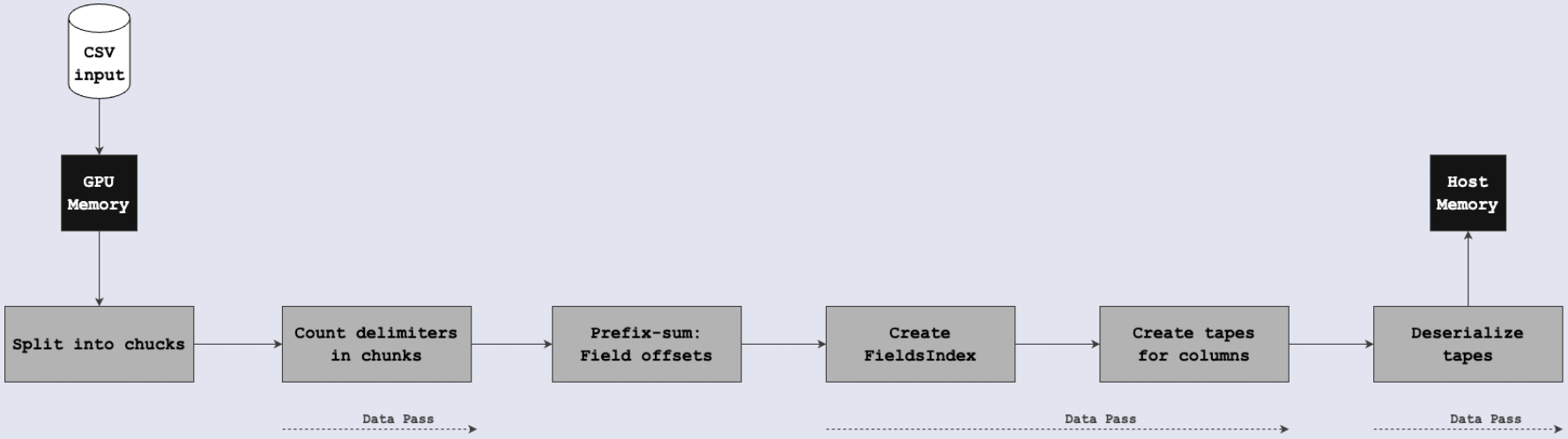


The GPU

- Around 80 SMs (streaming multicore processors)
- Threads within a warp can access the same Registers
- SIMD (Single Instruction Multiple Data) within a warp
- No branch prediction
- (Only) 32GB VRAM



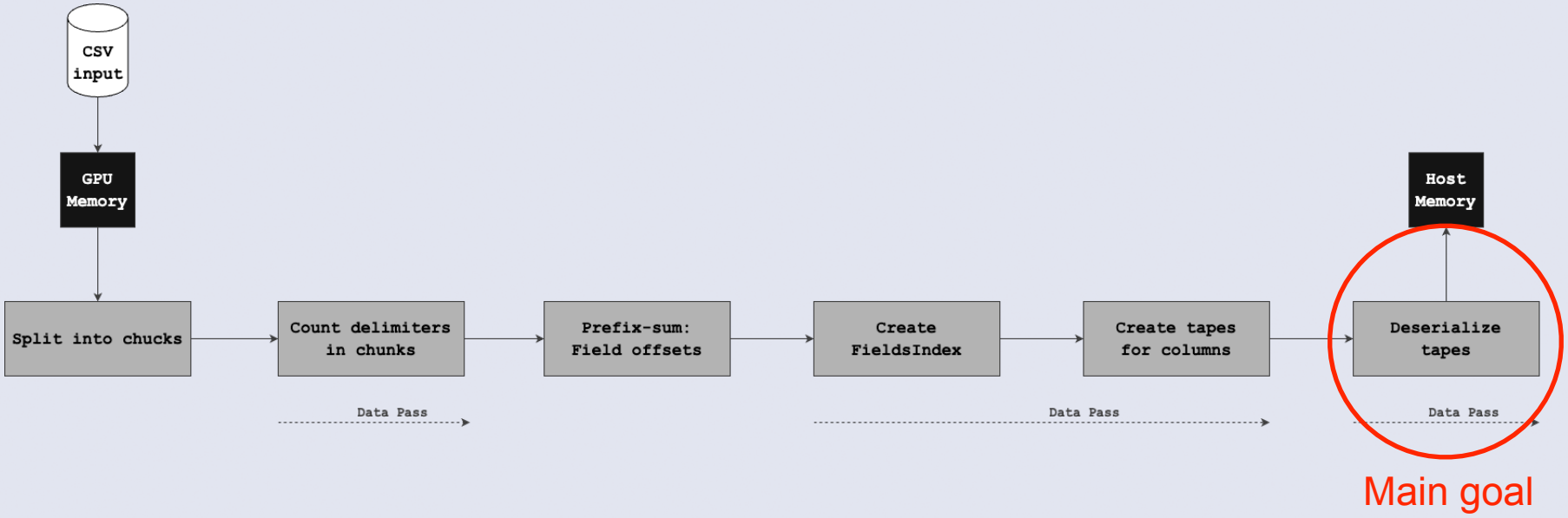
Concept of the CSV Parsing Algorithm - Overview



Parsers typically have complex control flow (warp divergence)

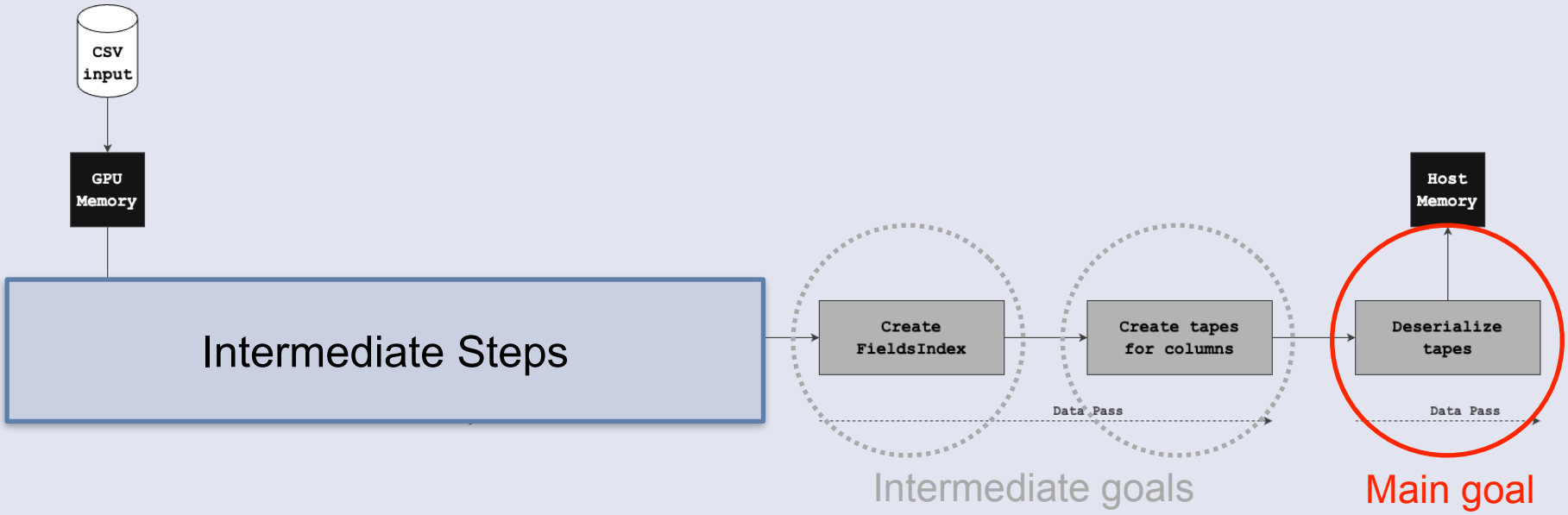
Main idea:
Simplify control flow at expense of additional data passes

Concept of the CSV Parsing Algorithm - Overview



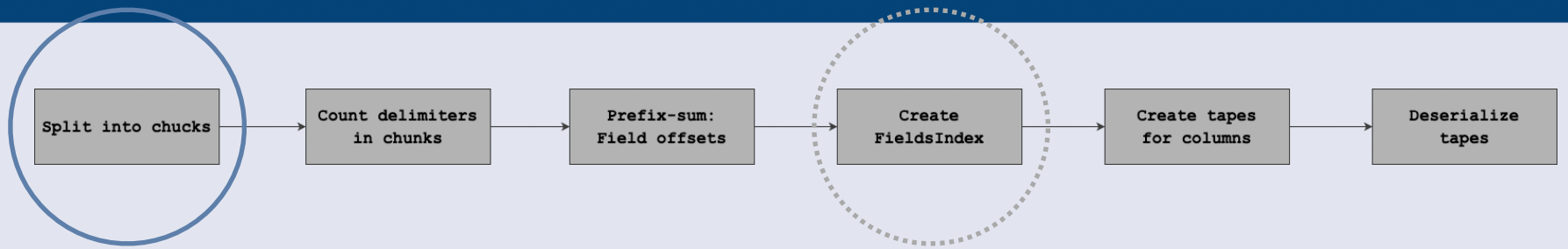
Deserialize: „Converting“ the data into concrete data structures

Concept of the CSV Parsing Algorithm - Overview



Tapes are here to exploit the fact that there is usually only one datatype in a column.

Which field belongs to which column? ⟶ FieldsIndex-Array




```
ID,Country,ISOCode,Population\n
1,Germany,DE,83149319\n
2,Spain,ES,47007367\n
3,France,FR,67076431\n
4,Italy,IT,60317116\n
5,Japan,JP,126150745\n
6,China,CN,1427647786\n
7,United States of America,US,328239523\n
8,Canada,CA,37894799\n
...
```

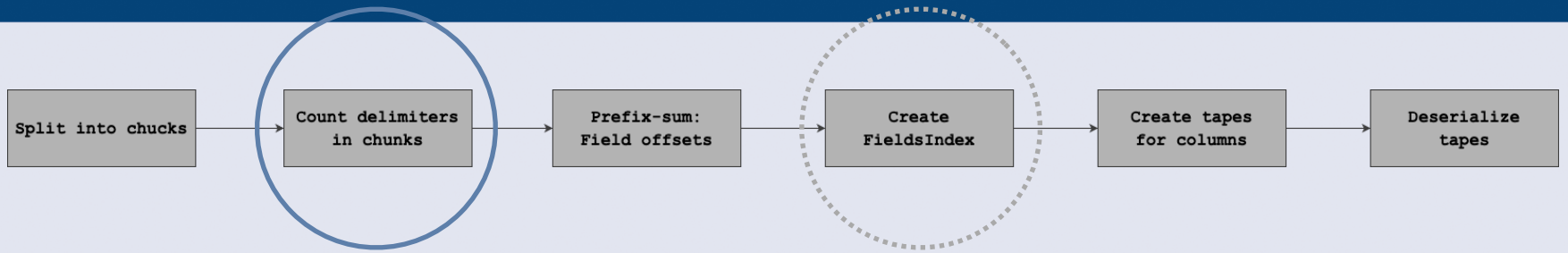
To exploit parallelization. One chunk per warp

chunk 0
```
ID,Country,ISOCode,Population\n1,Germany,DE,83149319\n2,Spain,ES,4
```

chunk 1
```
7007367\n3,France,FR,67076431\n4,Italy,IT,60317116\n5,Japan,JP,1261
```

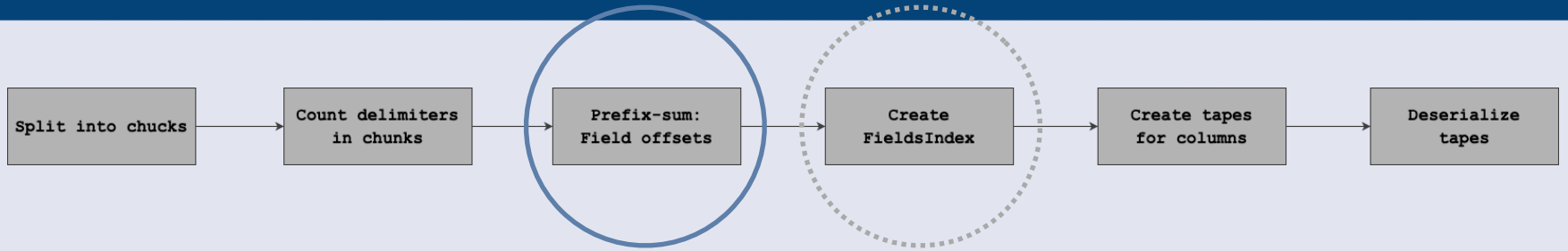
chunk 2
```
50745\n6,China,CN,1427647786\n7,United States of America,US,328239
```

chunk 3
```
523\n8,Canada,CA,37894799\n9,Australia,AU,25721892\n10,Russian F...
```



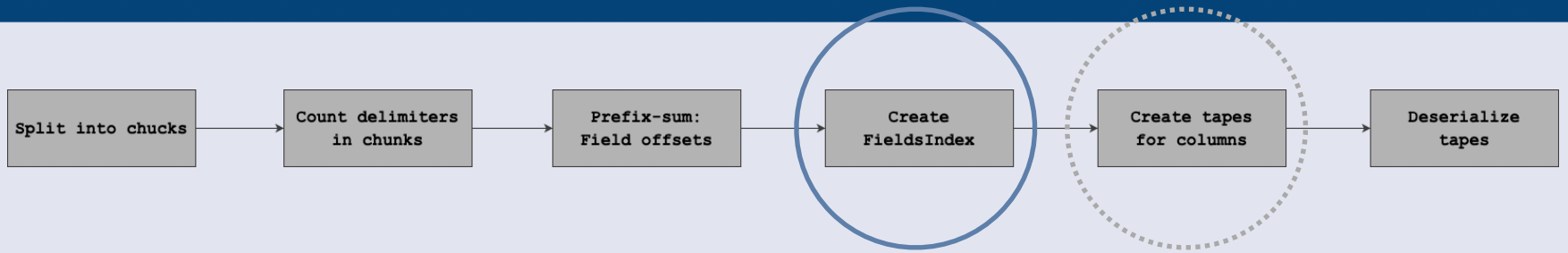
Chunk	Delimiter
chunk 0 ID,Country,ISOCode,Population\n1,Germany,DE,83149319\n2,Spain,ES,4	11
chunk 1 7007367\n3,France,FR,67076431\n4,Italy,IT,60317116\n5,Japan,JP,1261	12
chunk 2 50745\n6,China,CN,1427647786\n7,United States of America,US,328239	8
chunk 3 523\n8,Canada,CA,37894799\n9,Australia,AU,25721892\n10,Russian F...	10

Data pass 1



Chunk	Delimiter	Prefix Sum
chunk 0: ID,Country,ISOCode,Population\n1,Germany,DE,83149319\n2,Spain,ES,4	11	0
chunk 1: 7007367\n3,France,FR,67076431\n4,Italy,IT,60317116\n5,Japan,JP,1261	12	11
chunk 2: 50745\n6,China,CN,1427647786\n7,United States of America,US,328239	8	23
chunk 3: 523\n8,Canada,CA,37894799\n9,Australia,AU,25721892\n10,Russian F...	10	31

No data pass needed, just an array-operation.



Prefix Sum

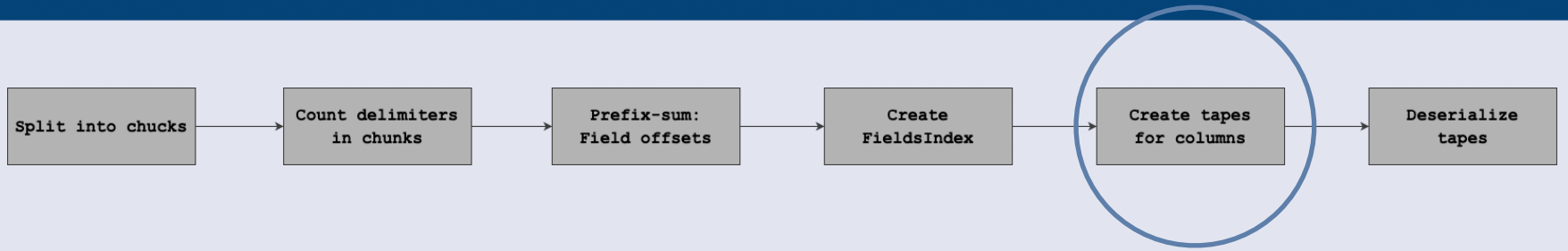
Chunk

0	+	ID,Country,ISOCode,Population\n1,Germany,DE,83149319\n2,Spain,ES,4
11	+	7007367\n3,France,FR,67076431\n4,Italy,IT,60317116\n5,Japan,JP,1261
23	+	50745\n6,China,CN,1427647786\n7,United States of America,US,328239
31	+	523\n8,Canada,CA,37894799\n9,Australia,AU,25721892\n10,Russian F...

FieldsIndex

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	11	19	30	32	40	43	52	54	60	63	72	74	81	84

Data pass 2




```
Disease,Year,Country,Cases\n
COVID-19,2020,DE,184492\n
H1N1,2009,ES,1538\n
Spanish Flu,1918,FR,187500\n
...
```

Disease . char(12)

C	O	V	I	D	-	1	9	\0	\0	\0	\0
H	1	N	1	\0	\0	\0	\0	\0	\0	\0	\0
S	p	a	n	i	s	h		F	l	u	\0

Year . uint(4)

2	0	2	0
2	0	0	9
1	9	1	8

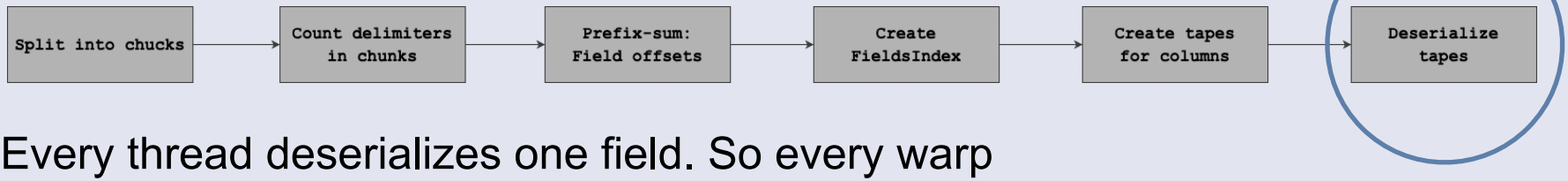
Country char(2)

D	E
E	S
F	R

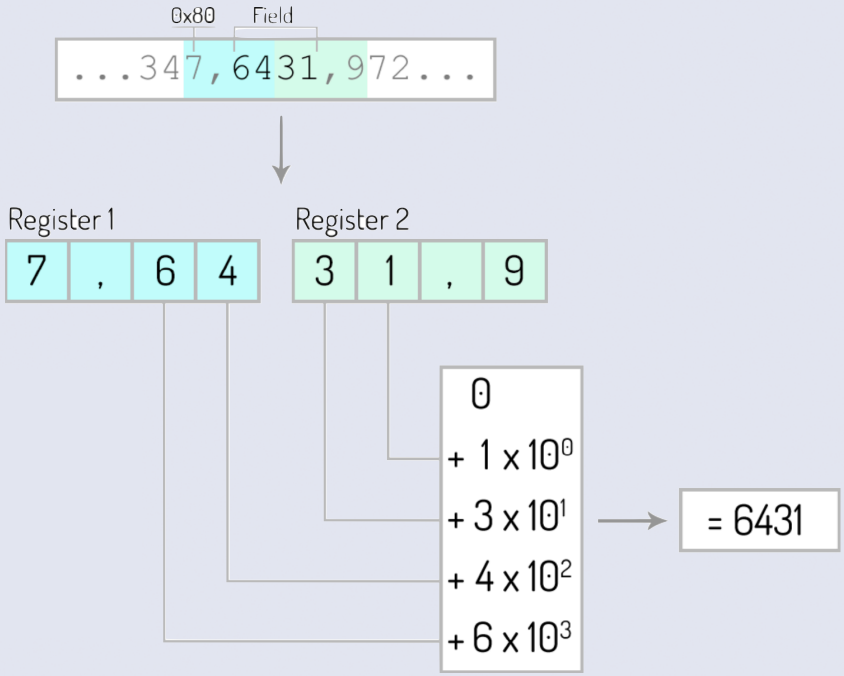
Cases uint(8)

1	8	4	4	9	2	\0	\0
1	5	3	8	\0	\0	\0	\0
1	8	7	5	0	0	\0	\0

Data pass 2

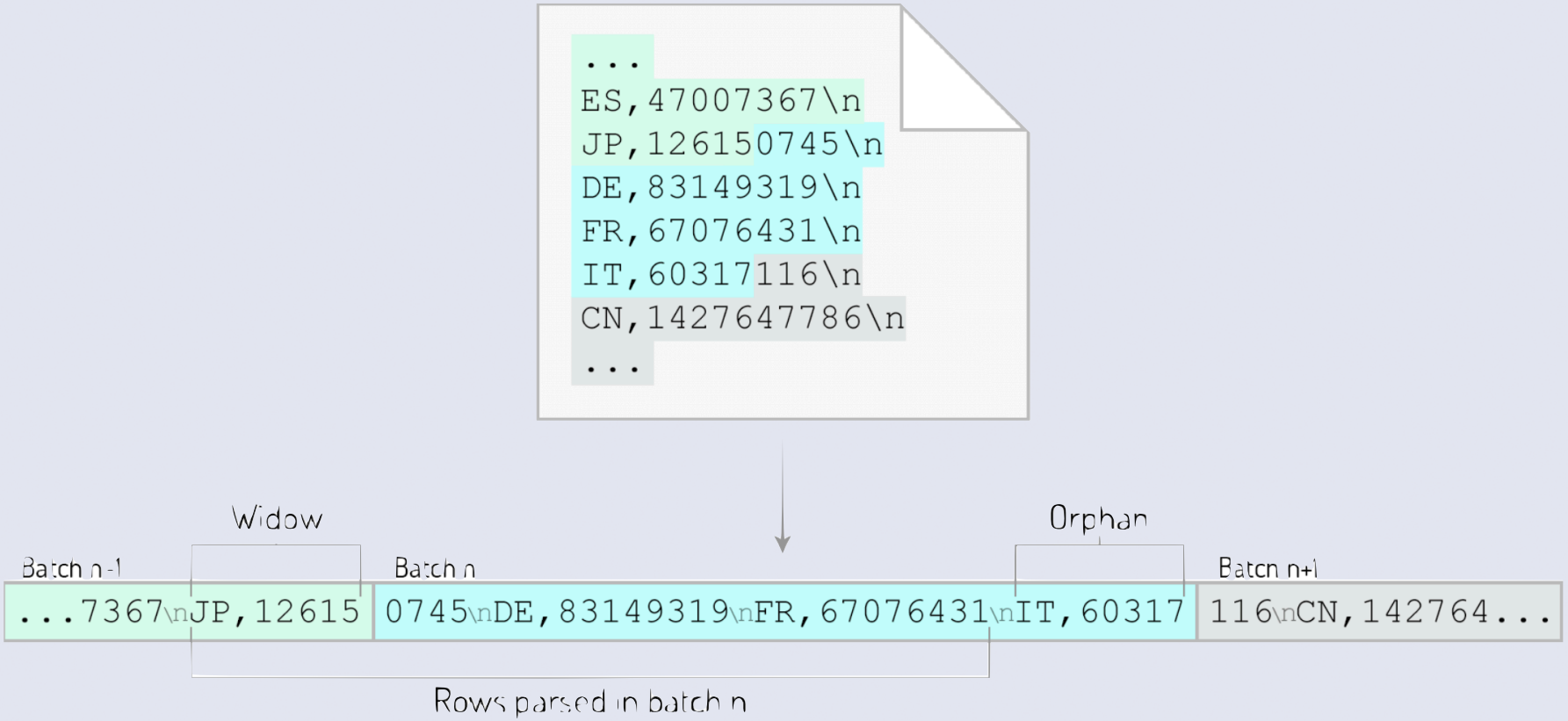


Every thread deserializes one field. So every warp deserializes 32 fields in parallel

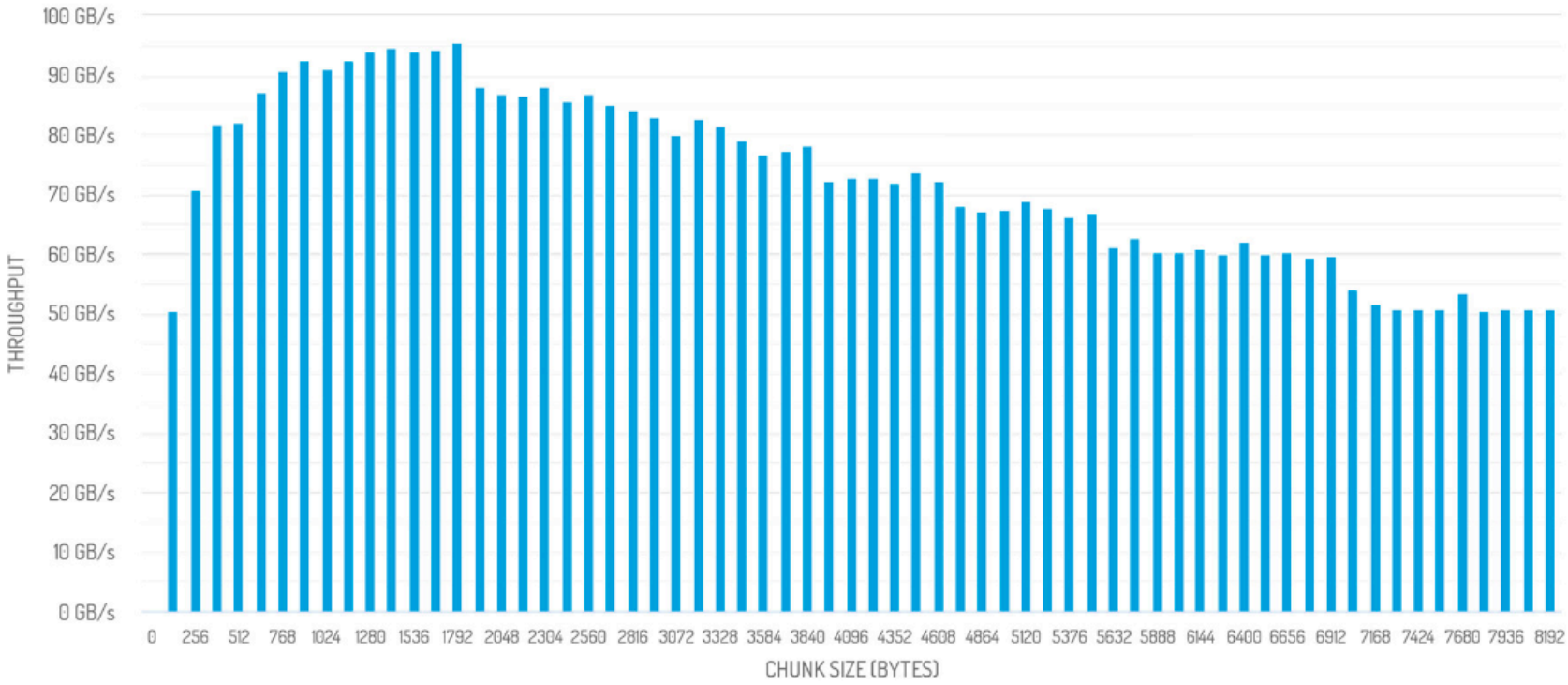


Data pass 3

Streaming

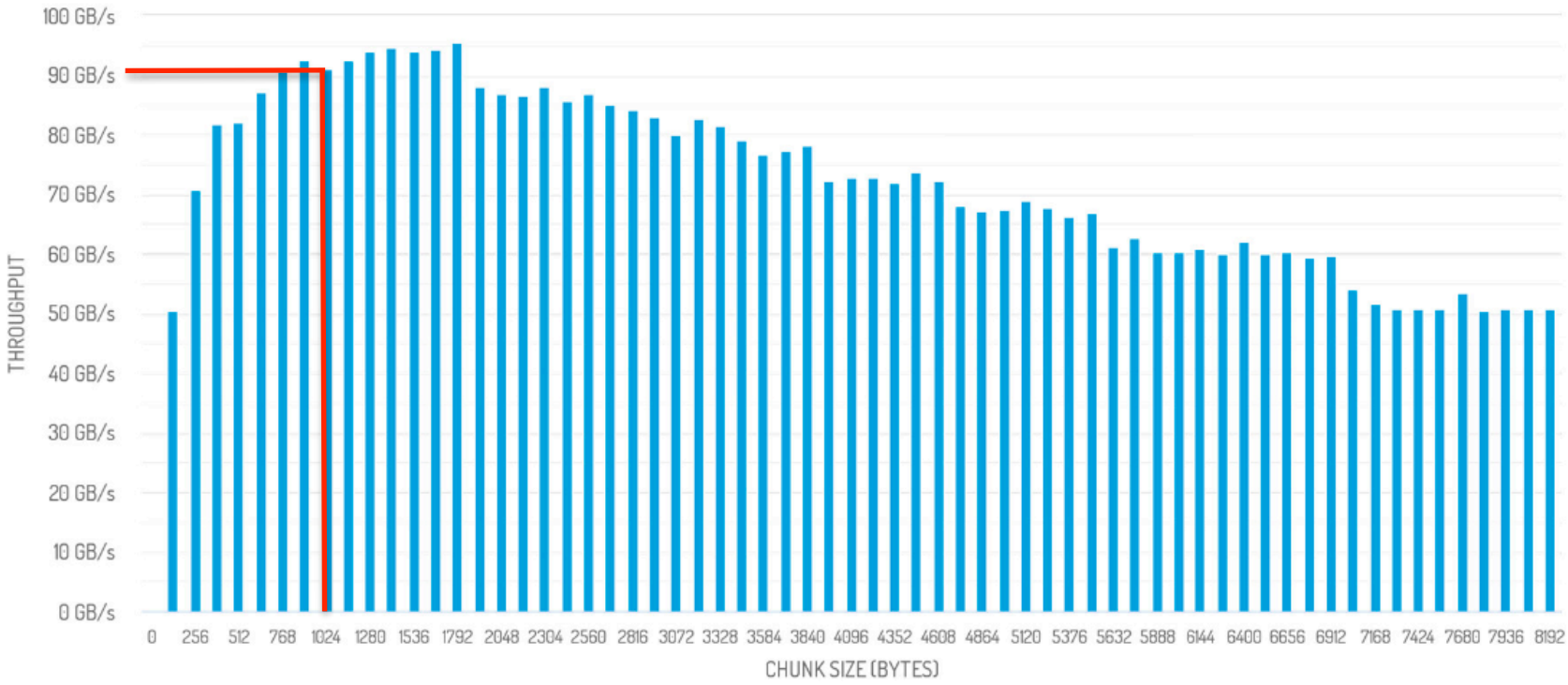

...
ES,47007367\n
JP,126150745\n
DE,83149319\n
FR,67076431\n
IT,60317116\n
CN,1427647786\n
...
Widow
Orphan
Batch n-1
Batch n
Batch n+1
...7367\nJP,12615
0745\nDE,83149319\nFR,67076431\nIT,60317
116\nCN,142764...
Rows parsed in batch n

Tuning Parameter: Chunk Size(Bytes)



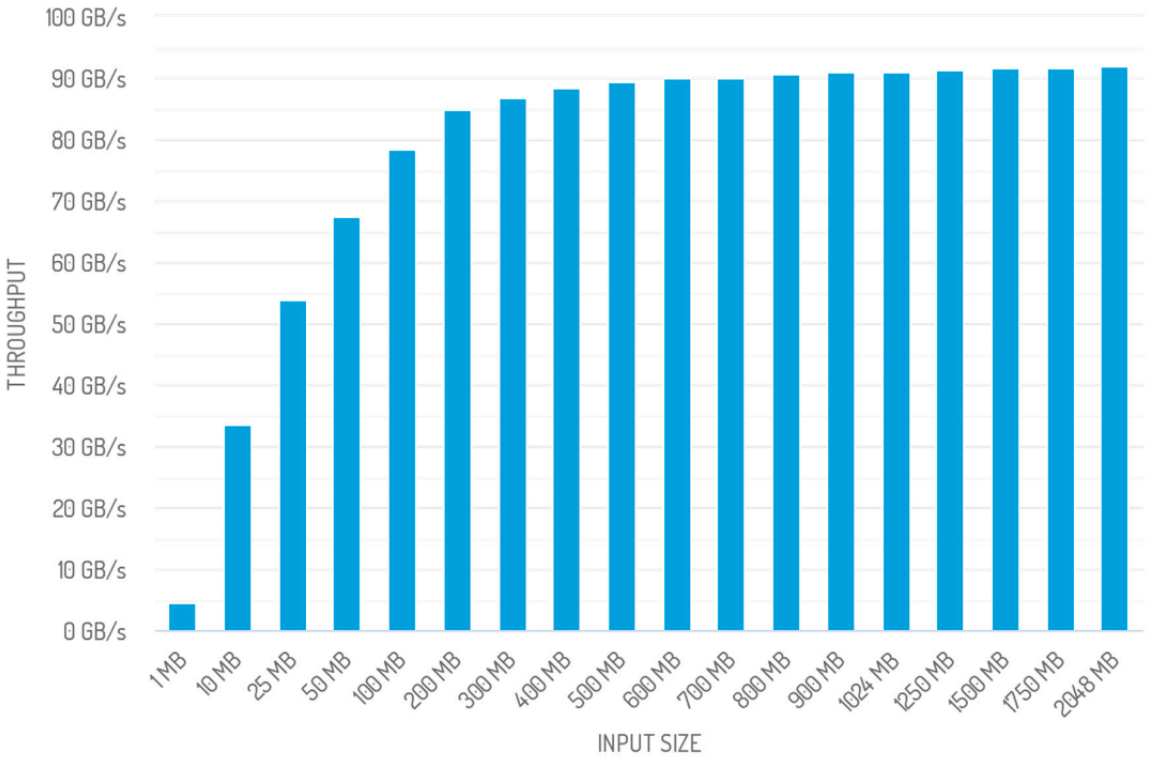
int_444 dataset: homogenous with 4 digits unsigned short values

Tuning Parameter: Chunk Size(Bytes)



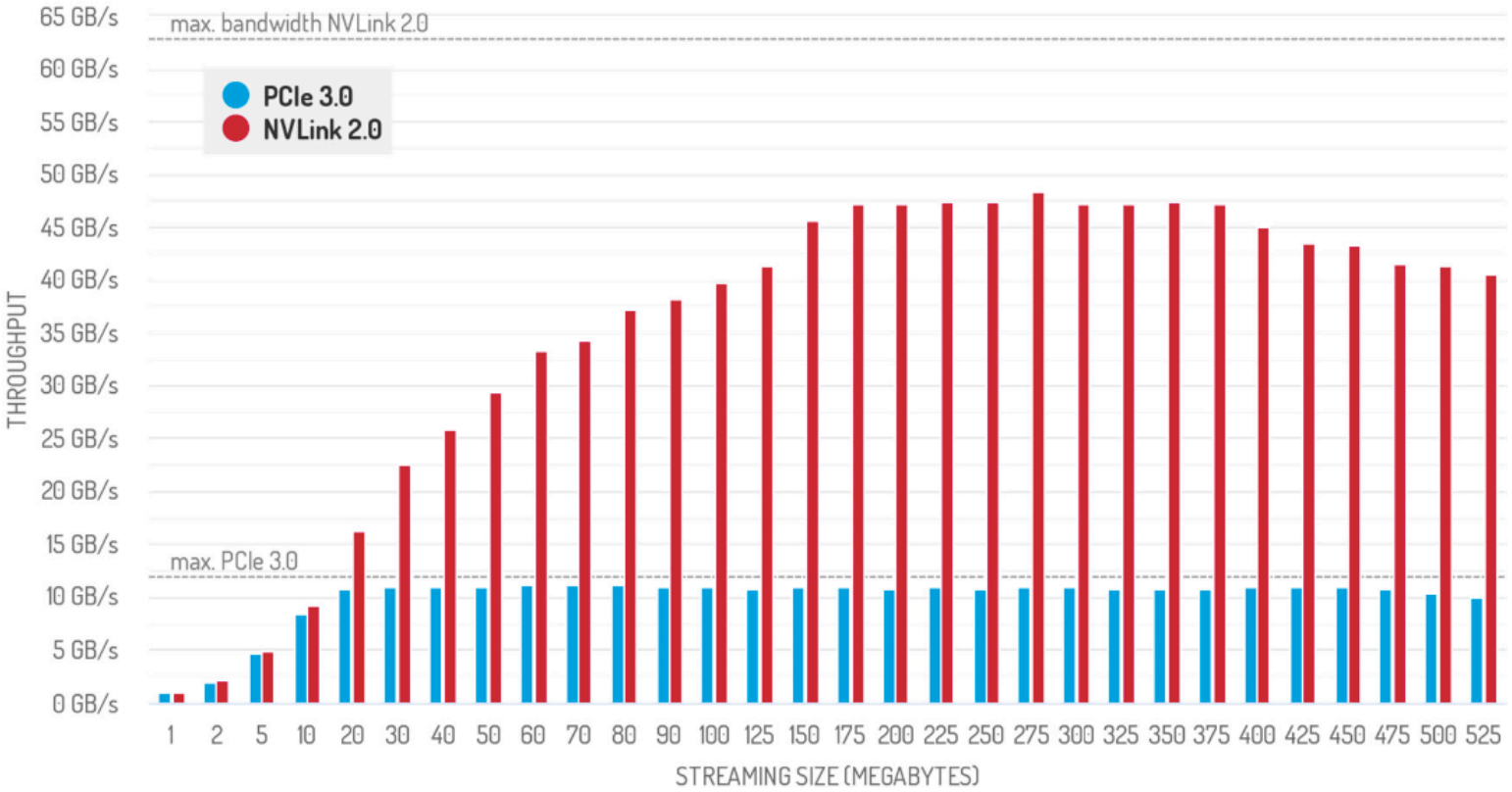
Sweet spot between: increasing resources per warp and reduce of overhead with scheduling, launching and processing

Tuning Parameter: Input Size



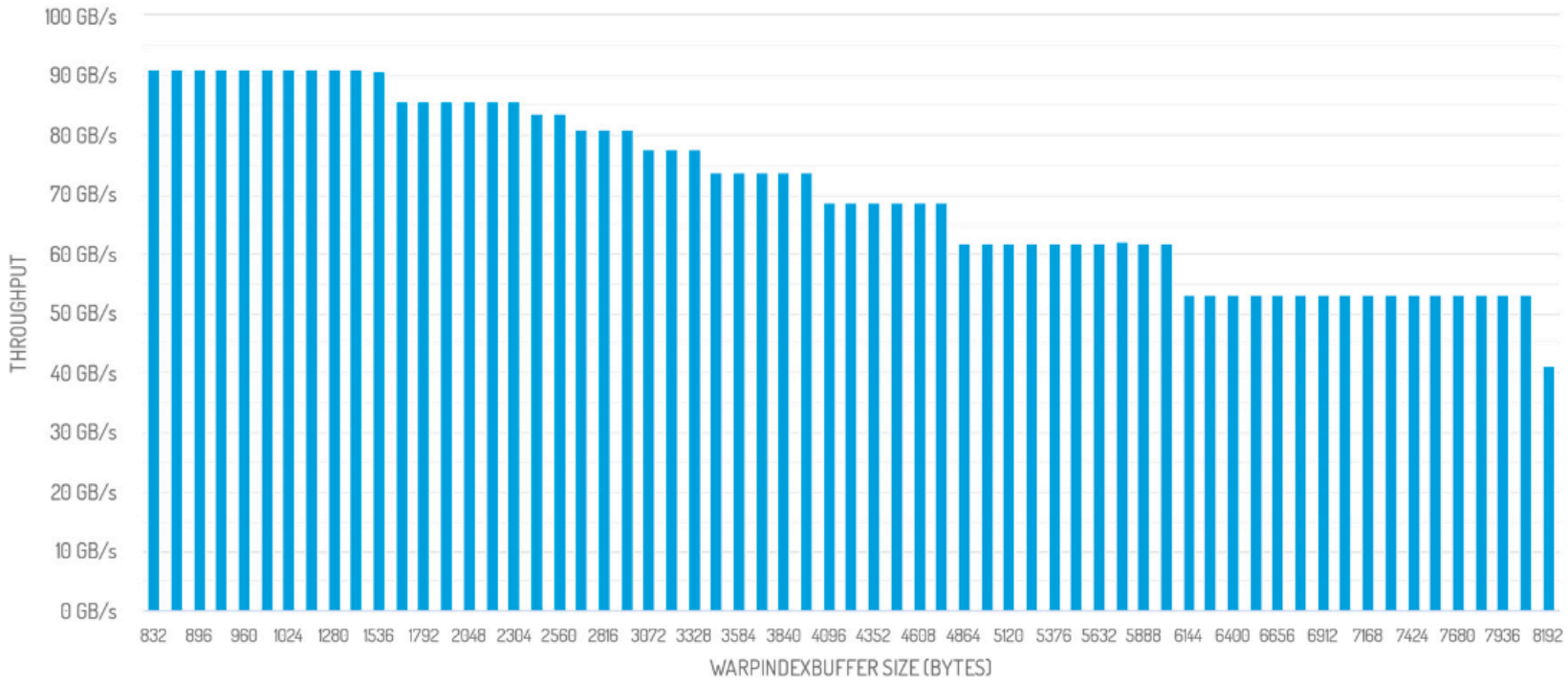
int_444 dataset: homogenous with 4 digits unsigned short values

Tuning Parameter: StreamingBatchSize



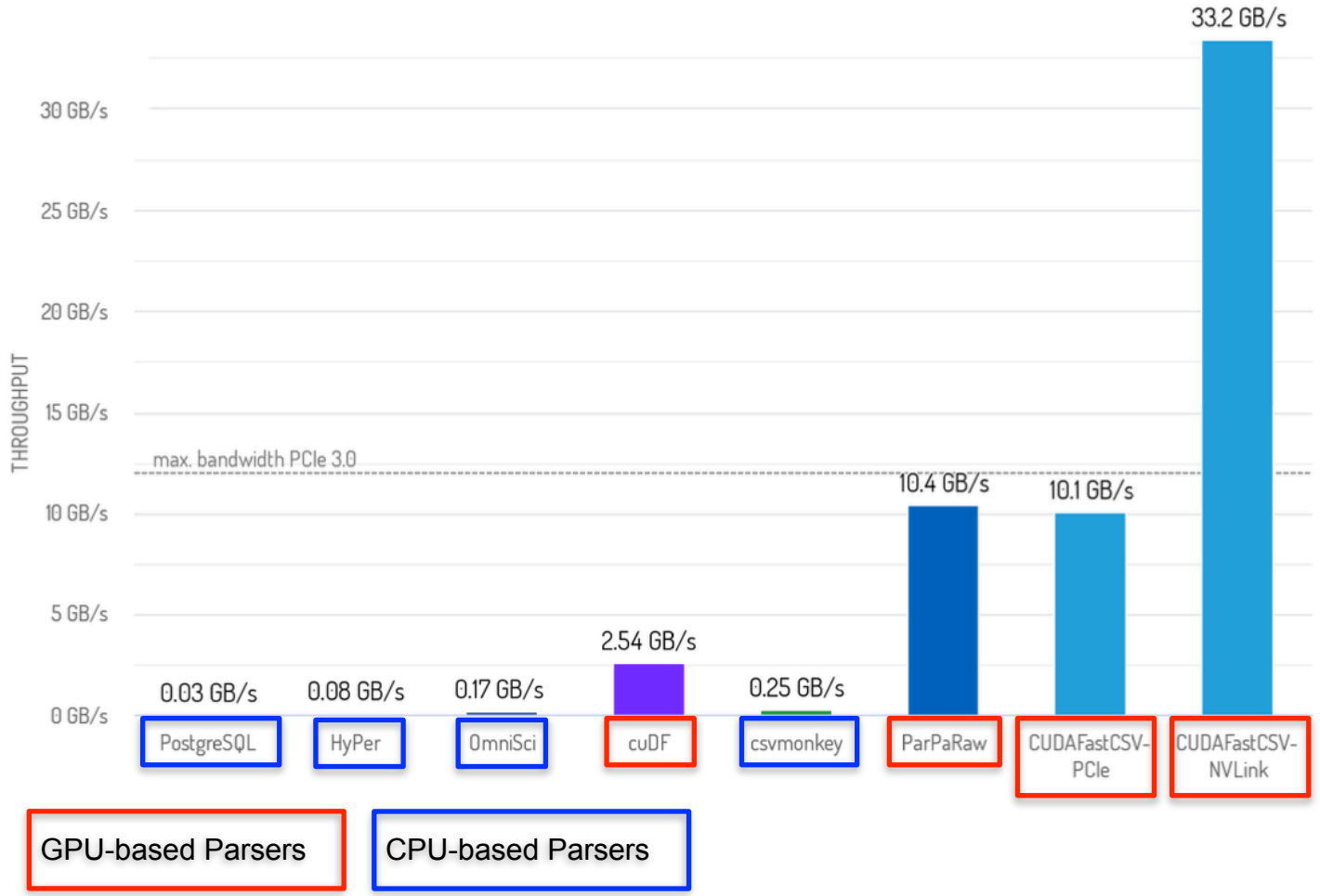
int_444 dataset: homogenous with 4 digits unsigned short values

Tuning Parameter: warpIndexBufferSize

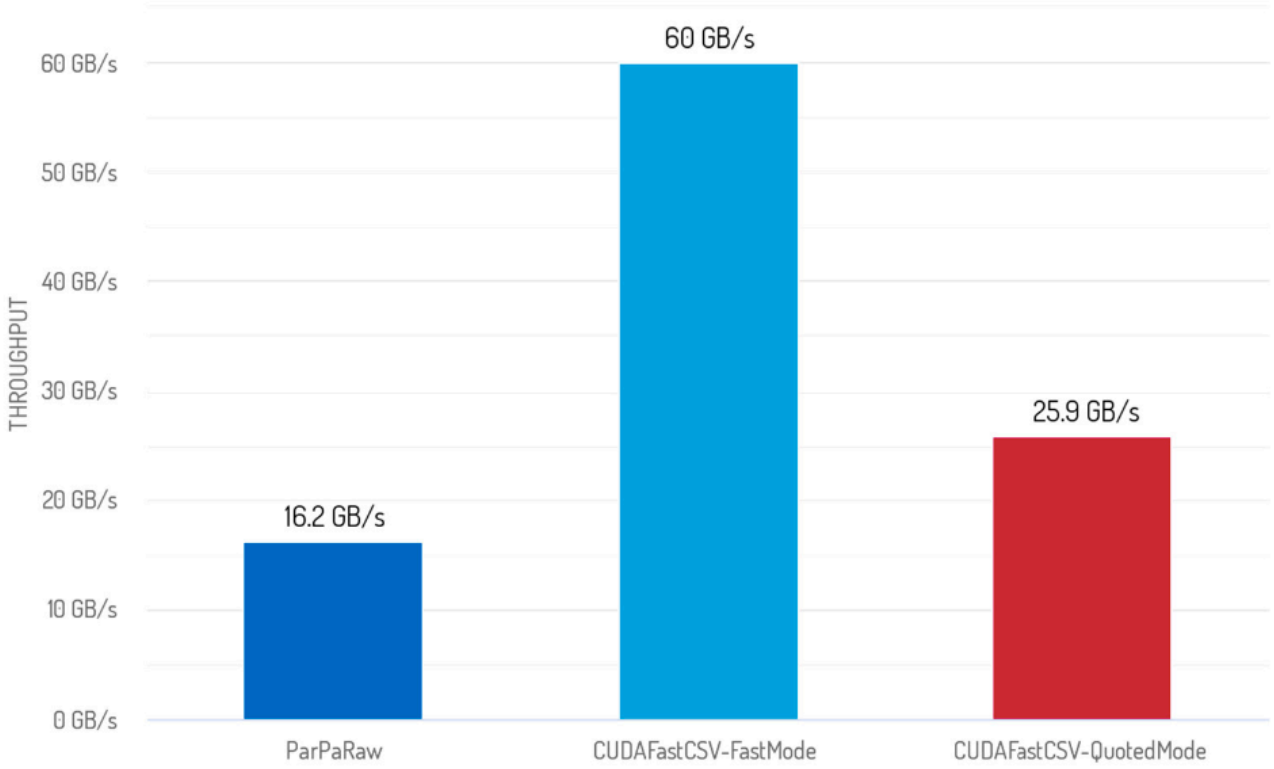


int_444 dataset: homogenous with 4 digits unsigned short values

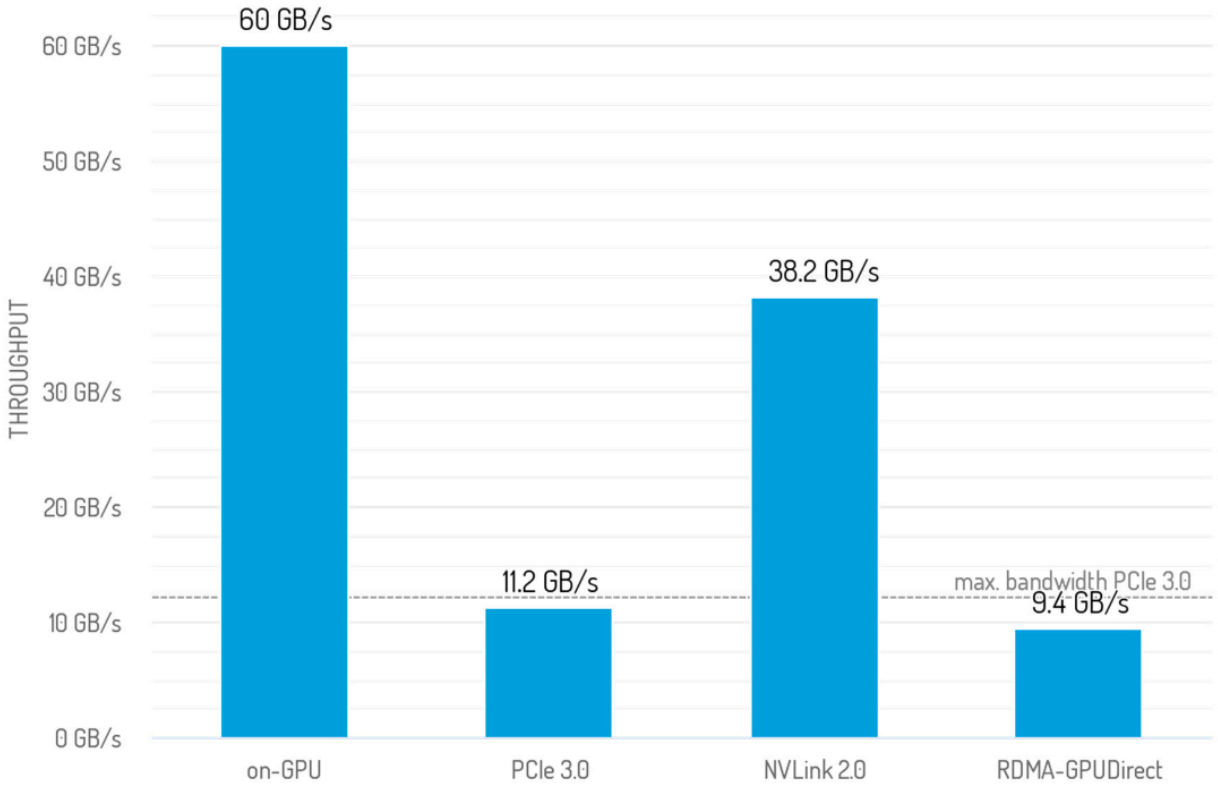
Results on NYC Yellow Taxi dataset



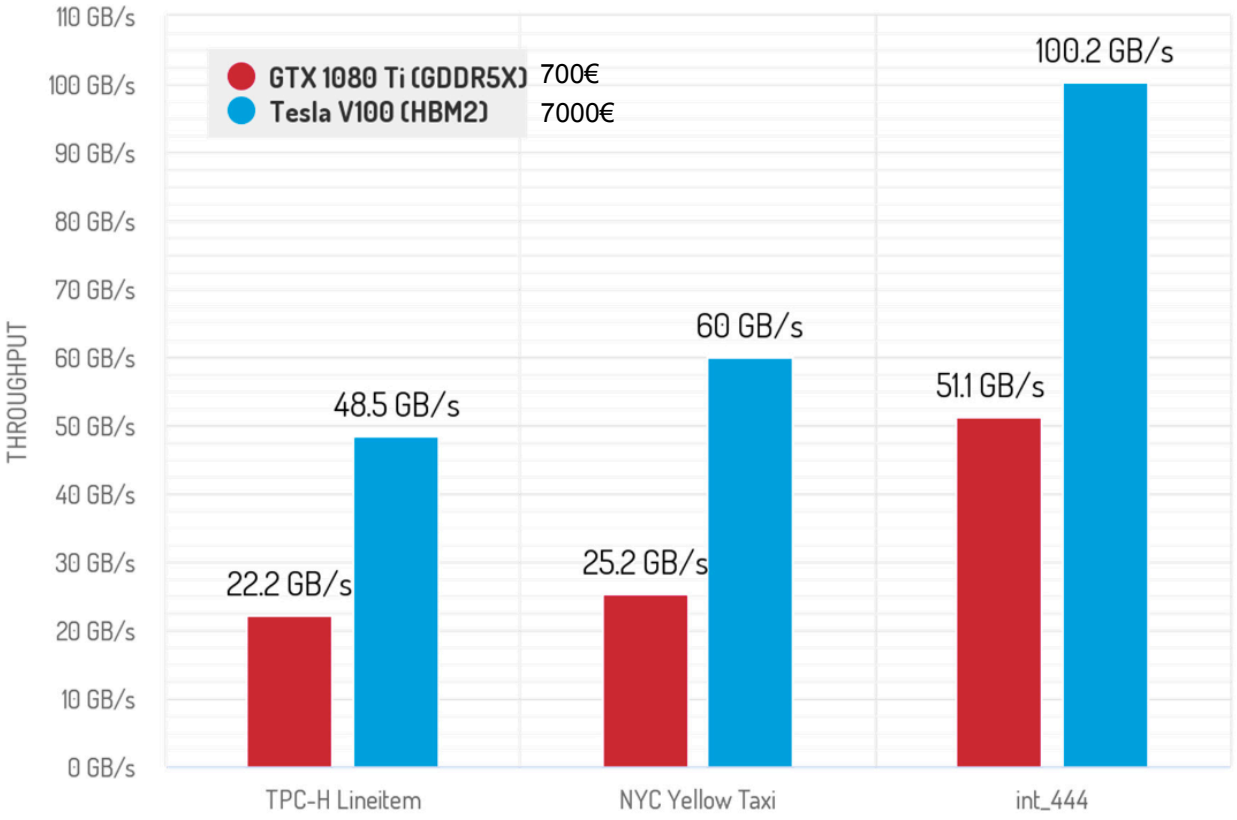
On-GPU Comparison



NYC Yellow Taxi - streaming performance



Comparison between desktop-grade GPU and server-grade GPU



Conclusion

- GPUs improve parsing performance
- PCIe 3.0 limits performance
- Network streaming is feasible
- GPUs can efficiently handle complex data formats
- GPUs facilitate data transformation (row-to-column)
- Desktop-grade GPUs provide good performance per cost