Database Compression on Graphics Processors

Seminar: Techniques for implementing main memory database systems

Maximilian Springer

26.11.2018

Why query processing on GPUs?

Why query processing on GPUs?

GPUs are very fast for **simple**, **highly-parallelizable**, **branchless** workflows.

Why query processing on GPUs? - Raw specifications

Specification	Intel® Core™ i9-7980XE[1]	GeForce RTX 2080 Ti[2]
Cores	18	4352

Why query processing on GPUs? - Raw specifications

Specification	Intel® Core™ i9-7980XE[1]	GeForce RTX 2080 Ti[2]
Cores	18	4352
Max Single Core Speed	4.20 GHz	1.635 GHz

Why query processing on GPUs? - Raw specifications

Specification	Intel® Core™ i9-7980XE[1]	GeForce RTX 2080 Ti[2]
Cores	18	4352
Max Single Core Speed	4.20 GHz	1.635 GHz
Max Mem Bandwidth	85.2 GB/s	616 GB/s

Why query processing on GPUs? - Raw specifications

Specification	Intel® Core™ i9-7980XE[1]	GeForce RTX 2080 Ti[2]
Cores	18	4352
Max Single Core Speed	4.20 GHz	1.635 GHz
Max Mem Bandwidth	85.2 GB/s	616 GB/s
GFLOPS	1152[3]	14700[4]

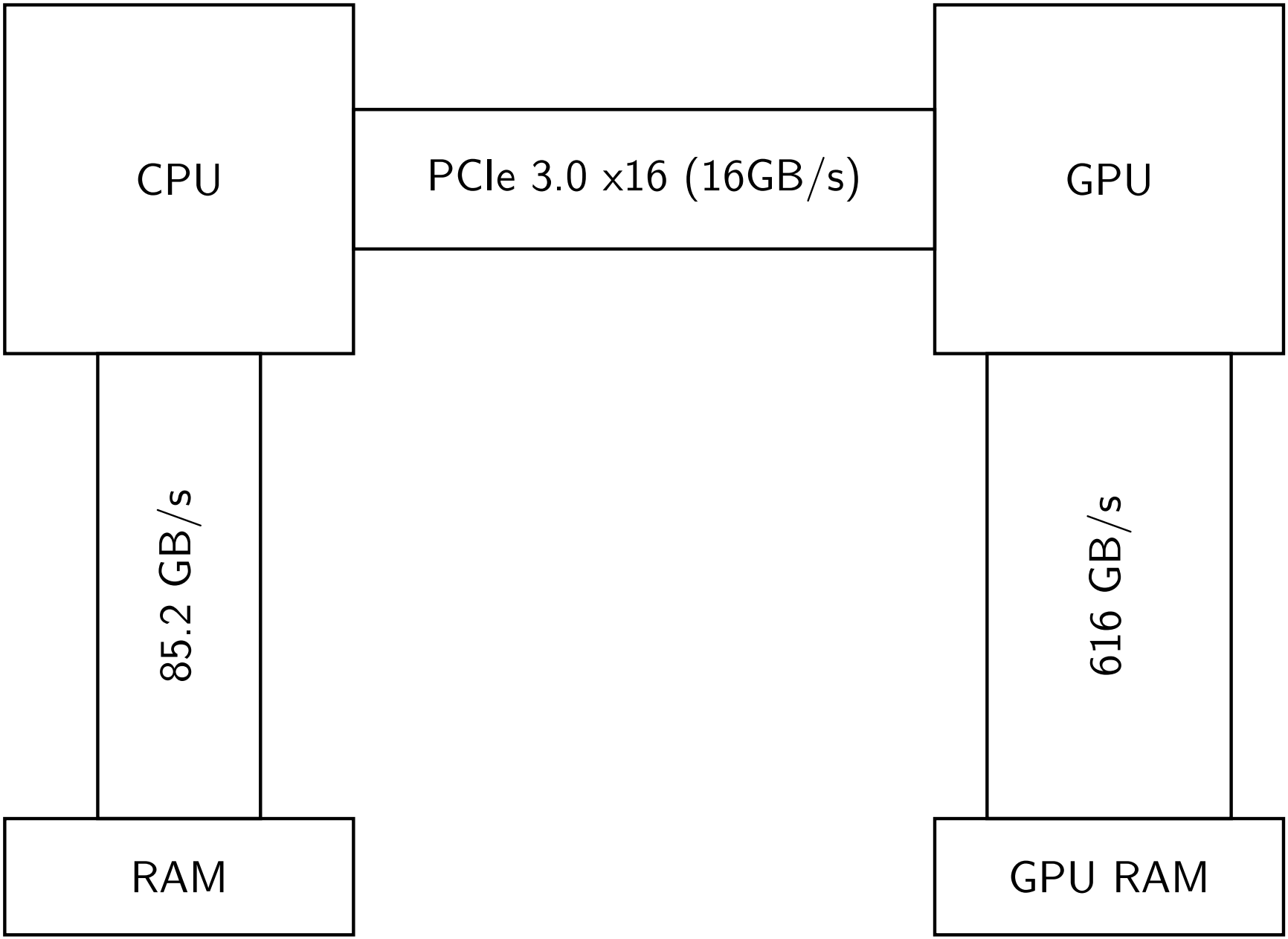
Why query processing on GPUs? - Raw specifications

Specification	Intel® Core™ i9-7980XE[1]	GeForce RTX 2080 Ti[2]
Cores	18	4352
Max Single Core Speed	4.20 GHz	1.635 GHz
Max Mem Bandwidth	85.2 GB/s	616 GB/s
GFLOPS	1152[3]	14700[4]
Max Mem Size	128 GB	11 GB

Why query processing on GPUs? - Drawbacks

- CPU $\Leftrightarrow$ GPU connection is slow

Why query processing on GPUs? - Drawbacks



Why query processing on GPUs? - Drawbacks

- CPU $\Leftrightarrow$ GPU connection is slow
- Single core speed is slower

Why query processing on GPUs? - Drawbacks

- CPU $\Leftrightarrow$ GPU connection is slow
- Single core speed is slower
- GPU code should be *branchless*, *simple* and *non-diverging*

Why query processing on GPUs? - Solutions to drawbacks

- Reduce copy cost necessary for query coprocessing

Why query processing on GPUs? - Solutions to drawbacks

- Reduce copy cost necessary for query coprocessing
- Idea: **calculation speed** $\Downarrow$ **copy speed** $\Uparrow$

Why query processing on GPUs? - Solutions to drawbacks

- Reduce copy cost necessary for query coprocessing
- Idea: **calculation speed** $\Downarrow$ **copy speed** $\Uparrow$
  - Transfer compressed data onto the GPU

Compression on GPU

- Focus on easy compression techniques

Compression on GPU

- Focus on easy compression techniques
  - Prevent branching

Compression on GPU

- Focus on easy compression techniques
  - Prevent branching
  - Simple code

Compression on GPU

- Focus on easy compression techniques
  - Prevent branching
  - Simple code
  - Less diverging between threads

Compression on GPU

- Focus on easy compression techniques
  - Prevent branching
  - Simple code
  - Less diverging between threads
- Chain them together to achieve better ratios

Compression techniques[5]

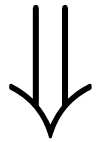
NS	Null Suppression with Fixed Length
NSV	Null Suppression with Variable Length
DICT	Dictionary
BITMAP	Bitmap
RLE	Run-Length-Encoding
DELTA	Delta

Cascaded compression

```
RLE, [[DELTA, NS] | NS]
```

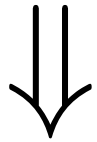
Cascaded compression

RLE, [[DELTA, NS] | NS]



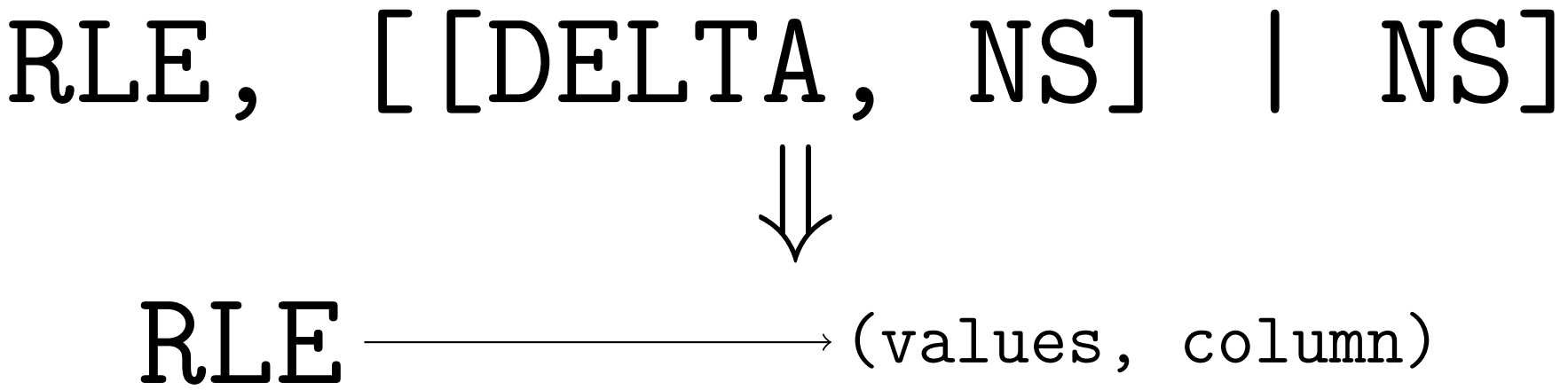
Cascaded compression

RLE, [[DELTA, NS] | NS]

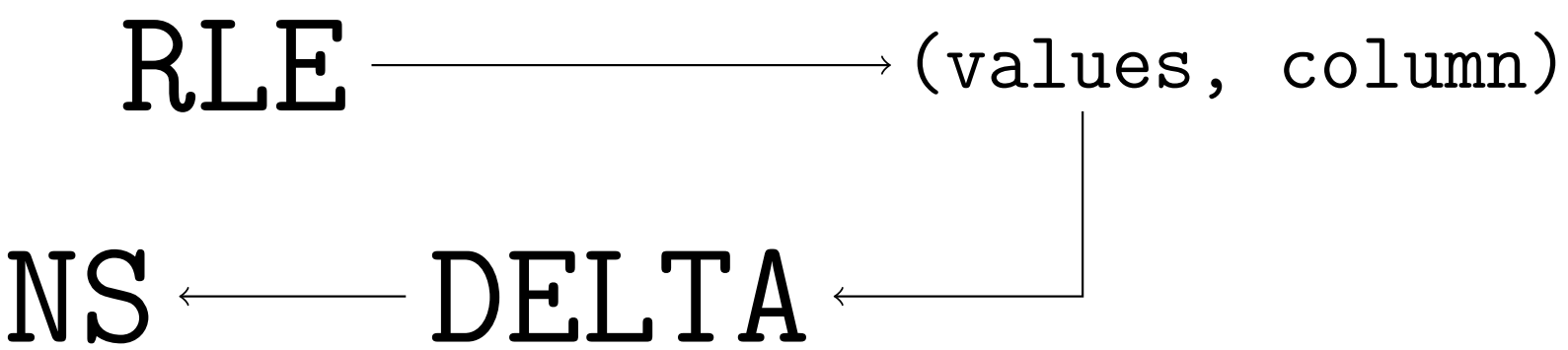


RLE

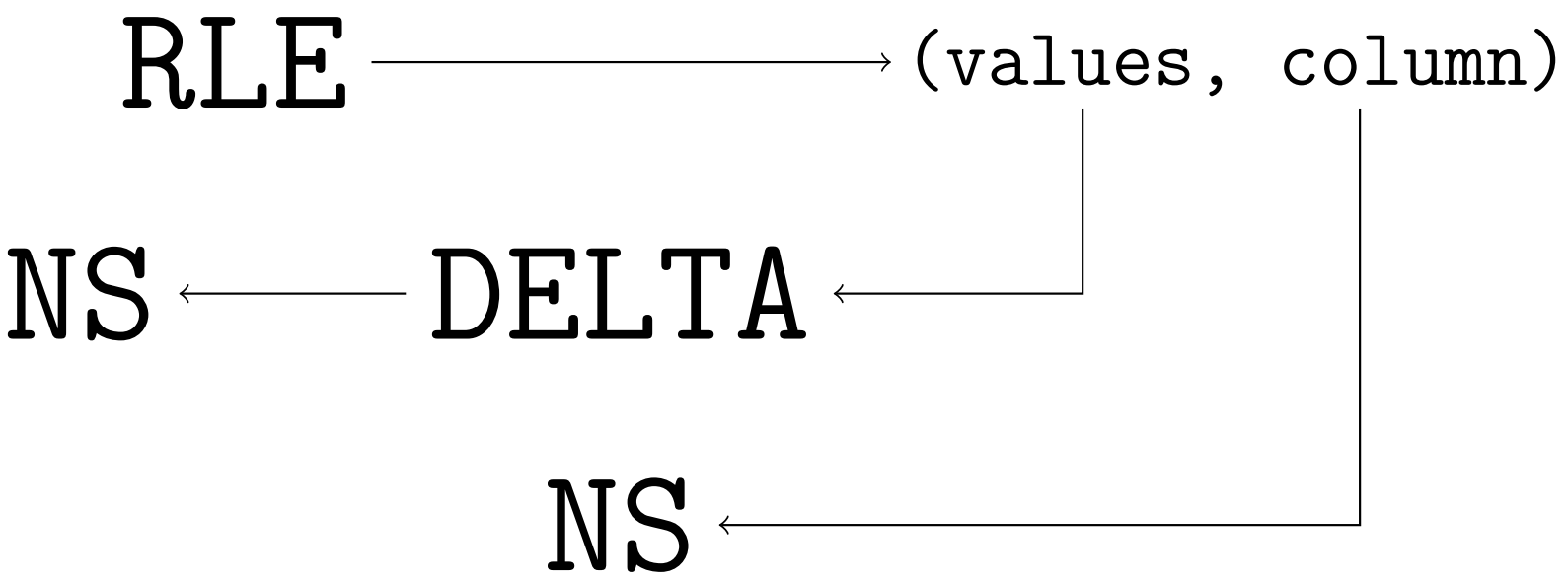
Cascaded compression



Cascaded compression



Cascaded compression

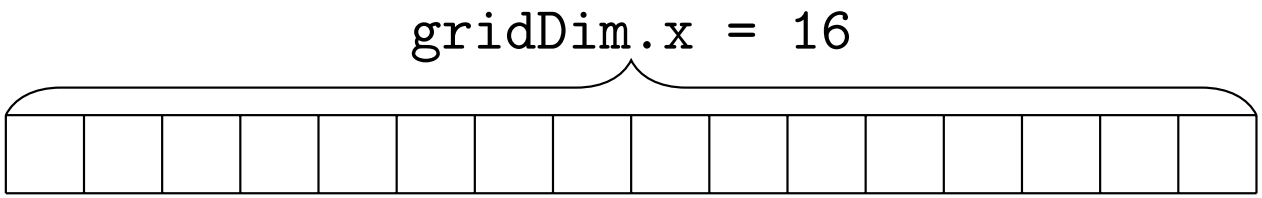


Architecture

GPU Architecture[6]

In CUDA the thread model is a **grid**

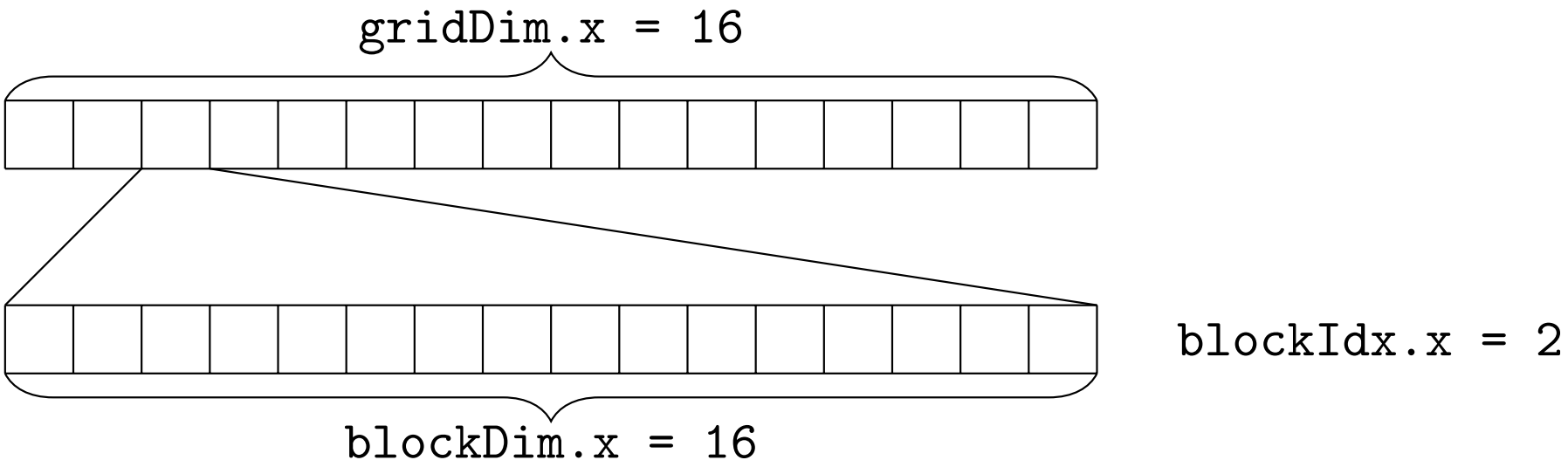
GPU Architecture[6]



GPU Architecture[6]

In CUDA the thread model is a **grid** of **blocks**

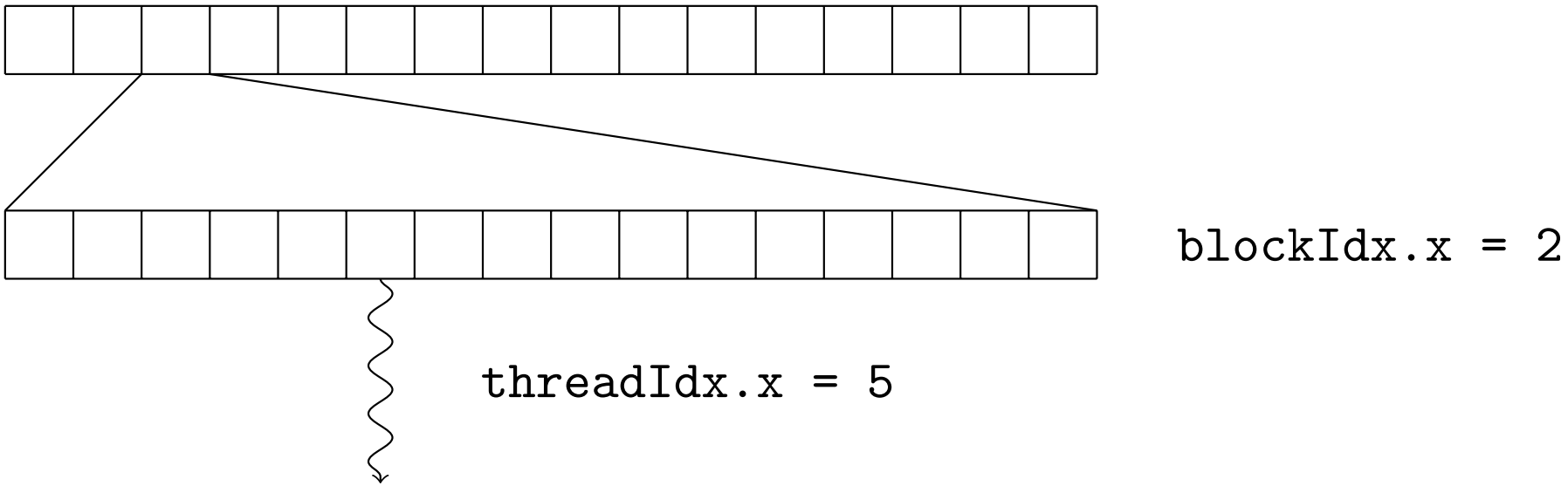
GPU Architecture[6]



GPU Architecture[6]

In CUDA the thread model is a **grid** of **blocks** of **threads**.

GPU Architecture[6]



GPU Architecture[6]

- Threads of a block organized in *warps* of 32 threads

GPU Architecture[6]

- Threads of a block organized in *warps* of 32 threads
- *Warps* executed on GPU processors

GPU Architecture[6]

- Threads of a block organized in *warps* of 32 threads
- *Warps* executed on GPU processors
- *Warp* code similar to *SIMD*

CUDA usage[6]

- Standard way: Compile CUDA code that can be invoked from host

CUDA usage[6]

- Standard way: Compile CUDA code that can be invoked from host
- JIT way: Use LLVM to compile CUDA on the fly[7][8]

CUDA usage[6]

- Standard way: Compile CUDA code that can be invoked from host
- JIT way: Use LLVM to compile CUDA on the fly[7][8]
  - Allows optimization of compression schema code

CUDA usage[6]

- Standard way: Compile CUDA code that can be invoked from host
- JIT way: Use LLVM to compile CUDA on the fly[7][8]
  - Allows optimization of compression schema code
  - Simplify combination of compression code

CUDA usage[6]

- Standard way: Compile CUDA code that can be invoked from host
- JIT way: Use LLVM to compile CUDA on the fly[7][8]
  - Allows optimization of compression schema code
  - Simplify combination of compression code
  - Allow custom optimization

Primitives

Primitives

- Map

Primitives

- Map
  - One thread per element in the array executing the map function

Primitives

- Map
  - One thread per element in the array executing the map function
- Scatter

Primitives

- Map
  - One thread per element in the array executing the map function
- Scatter
  - One thread per write position

Primitives

- Map
  - One thread per element in the array executing the map function
- Scatter
  - One thread per write position
- Prefix Sum[9]

Primitives

- Map
  - One thread per element in the array executing the map function
- Scatter
  - One thread per write position
- Prefix Sum[9]
  - Divide and conquer workload utilizing warp instructions and blocks

Dynamic Parallelism[6]

Dynamic Parallelism[6]

- Allows the launch of kernels of different size from another kernel

Dynamic Parallelism[6]

- Allows the launch of kernels of different size from another kernel
- Allows modelling of primitives via kernels

Dynamic Parallelism[6]

- Allows the launch of kernels of different size from another kernel
- Allows modelling of primitives via kernels
- Minimize wasted GPU processor time

Optimizations

Optimizations

- Merge kernels

Optimizations

- Merge kernels
- Create LLVM `Pass`[10] to check whether kernel invocation and kernels allow this

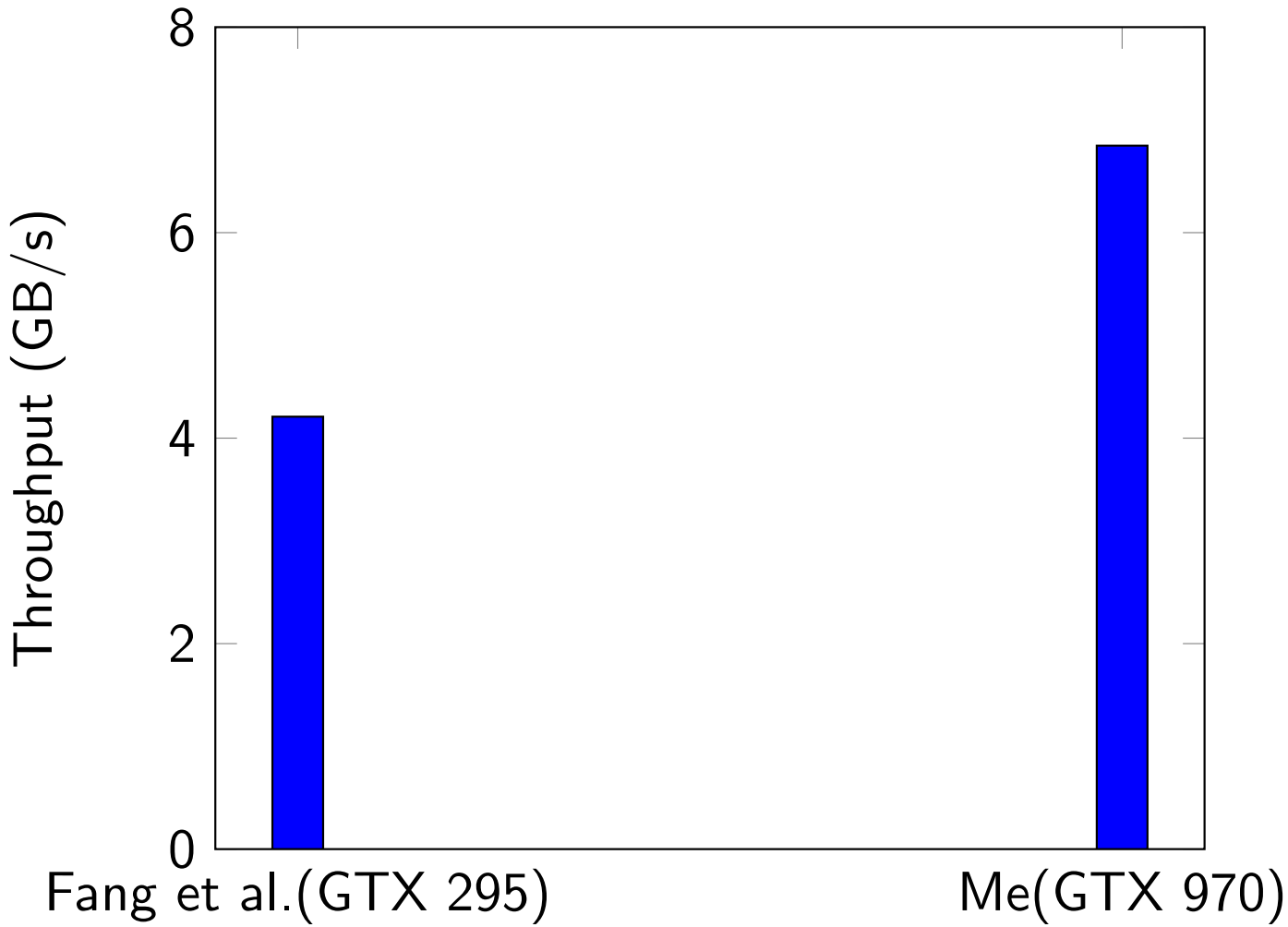
Optimizations

- Merge kernels
- Create LLVM `Pass`[10] to check whether kernel invocation and kernels allow this
- Combine frequents combined primitives together

Benchmarks

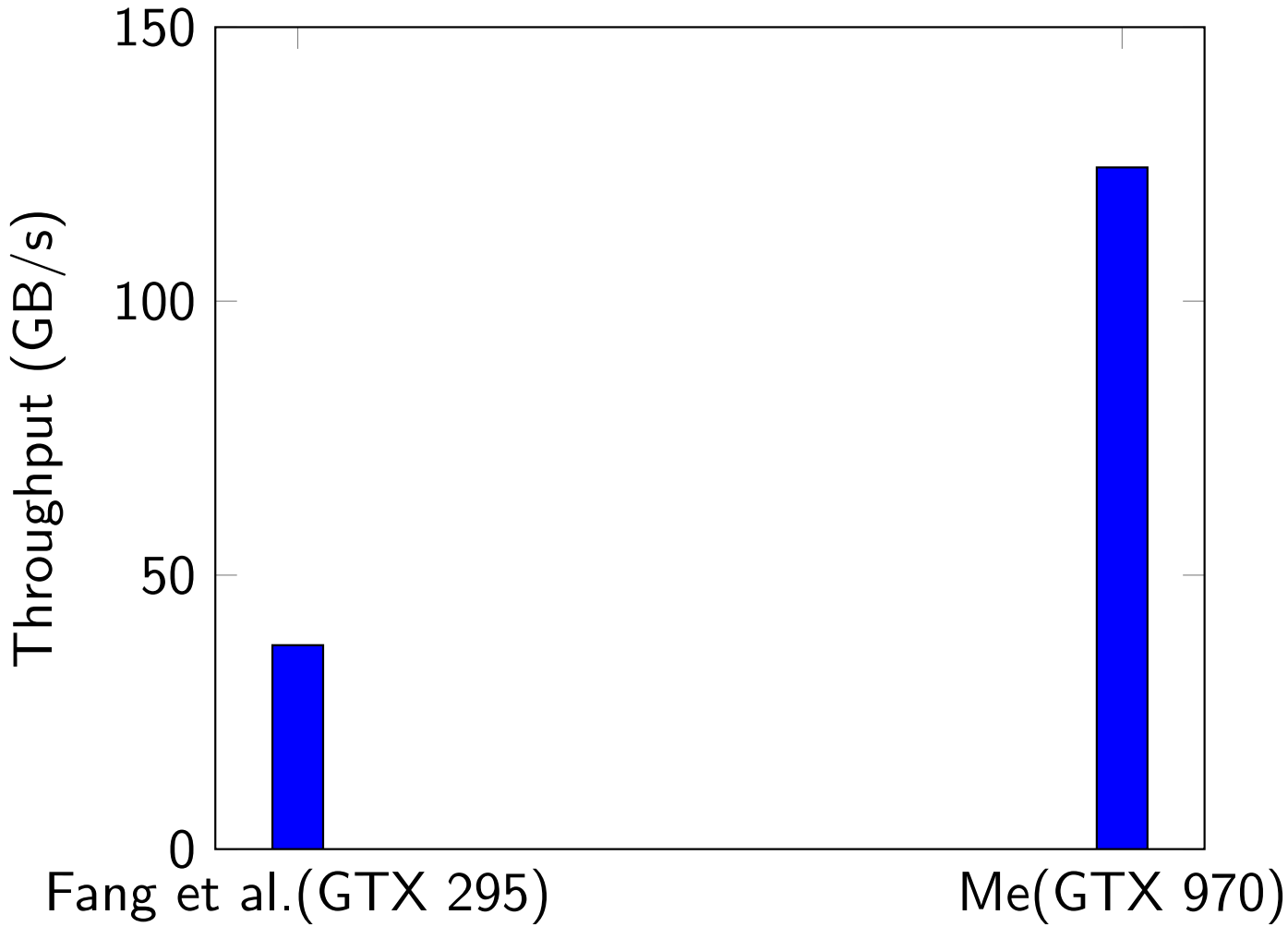
l_partkey(RLE(6.6%))

l_partkey column from TPC-H (scale = 10, 60M rows)



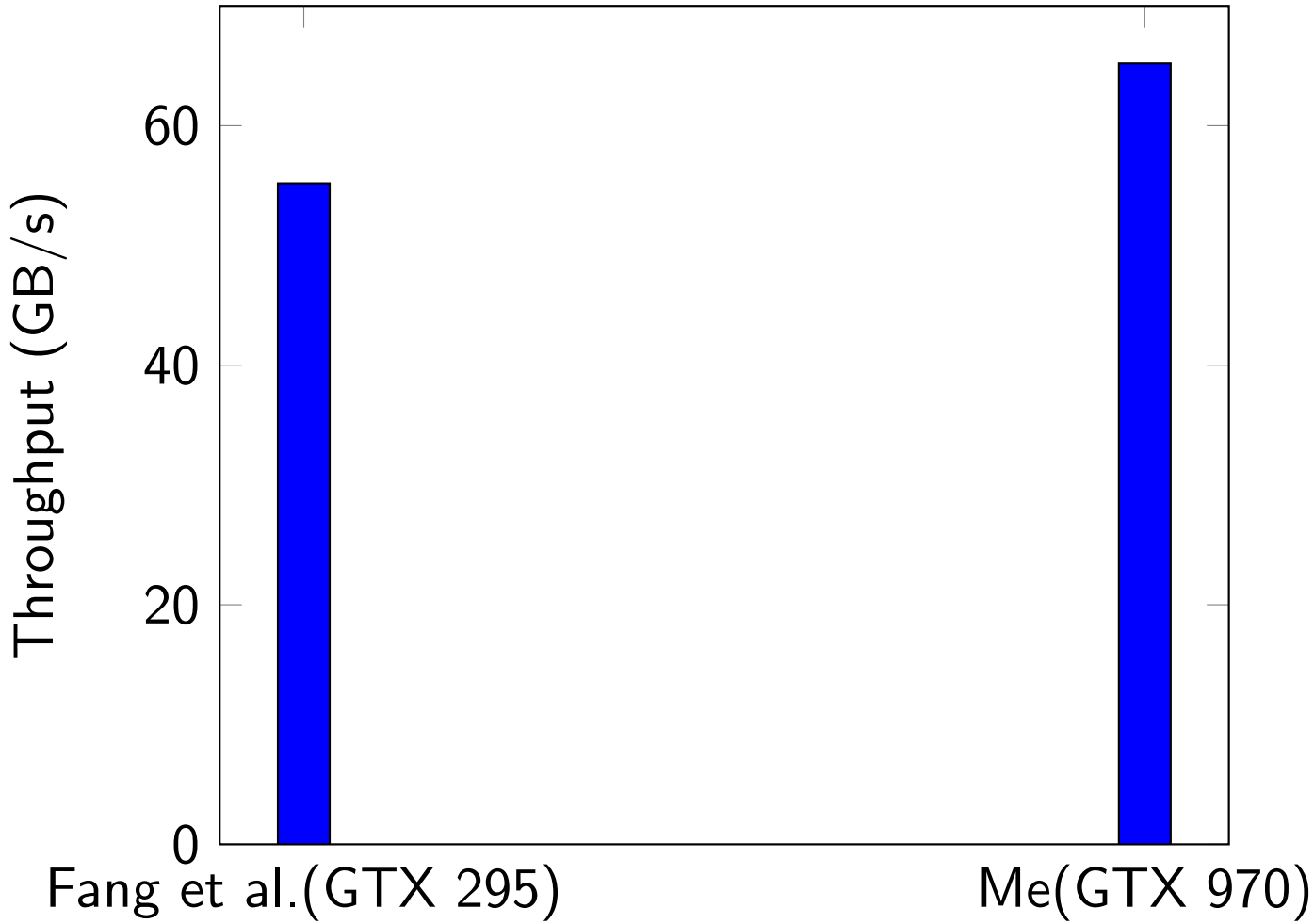
l_shipmode(DICT(50%))

l_shipmode column from TPC-H (scale = 10, 60M rows)



l_quantity(NS(75%))

l_quantity column from TPC-H (scale = 10, 60M rows)



References I

Intel Corporation. *Intel Core i9-7980XE Extreme Edition Processor*. URL: https://ark.intel.com/products/126699/Intel-Core-i9-7980XE-Extreme-Edition-Processor-24-75M-Cache-up-to-4-20-GHz- (visited on 11/20/2018).

NVIDIA Corporation. *GeForce RTX 2080 Ti Graphics Card — NVIDIA*. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/rtx-2080-ti/ (visited on 11/20/2018).

Intel Corporation. *Export Compliance Metrics for Intel® Microprocessors Intel Core® Processors*. URL: https://www.intel.com/content/dam/support/us/en/documents/processors/APP-for-Intel-Core-Processors.pdf (visited on 11/20/2018).

References II

Heise Medien. *Nvidia GeForce RTX 2080 und 2080 Ti: Alle Herstellerkarten im Überblick*. URL: https://www.heise.de/newsticker/meldung/Nvidia-GeForce-GTX-2080-und-2080-Ti-Alle-Herstellerkarten-im-Ueberblick-4142326.html (visited on 11/20/2018).

Wenbin Fang, Bingsheng He, and Qiong Luo. "Database compression on graphics processors". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 670–680.

NVIDIA Corporation. *NVIDIA CUDA Programming Guide*. URL: http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf (visited on 11/19/2018).

LLVM Project. *User Guide for NVPTX Backend — LLVM 8 documentation*. URL: https://llvm.org/docs/NVPTXUsage.html (visited on 11/19/2018).

References III

NVIDIA Corporation. *NVVM IR :: CUDA Toolkit Documentation*. URL: https://docs.nvidia.com/cuda/nvvm-ir-spec/index.html (visited on 11/19/2018).

W Daniel Hillis and Guy L Steele Jr. "Data parallel algorithms". In: *Communications of the ACM* 29.12 (1986), pp. 1170–1183.

LLVM Project. *Writing an LLVM Pass — LLVM 8 documentation*. URL: https://llvm.org/docs/WritingAnLLVMPass.html (visited on 11/19/2018).

Thank You!
Questions?