CS 764: Topics in Database Management Systems
Lecture 27: GPU Databases

Xiangyao Yu
12/7/2022
Announcements

DAWN workshop
– Reserve a presentation slot using the following google sheet
  https://docs.google.com/spreadsheets/d/1Re1M9FmJwl_YkidhNgeV0iKncIcslsFrK_J1PMidaAuw/edit?usp=sharing
– 8-min per group (presentation + QA)

Project report (DDL: Dec. 19)
– Submit to the hotcrp website (like the proposal)

Submit course evaluation on aefis.wisc.edu
System Architecture
Advantages of GPU for Data Analytics

Advantage 1: **High computational power**
- GPU has massive parallelism using SIMT model
Advantages of GPU for Data Analytics

Advantage 1: **High computational power**

Advantage 2: **Higher Memory Bandwidth**

– GPU memory bandwidth is one-order-of-magnitude higher than CPU memory bandwidth
Challenges of GPU for Data Analytics

Challenge 1: **Limited memory capacity**
- Some data sets do not fit in GPU memory
Challenges of GPU for Data Analytics

Challenge 1: **Limited memory capacity**
Challenge 2: **Limited interconnect bandwidth**
   – Inter-device data transfer is a performance bottleneck
GPU Database Roadmap

Performance

Data size

CPU memory capacity (100GB-10TB)

CPU DB
GPU Database Roadmap

Performance

Data size

GPU memory capacity
(8-80 GB)

CPU memory capacity
(100GB-10TB)

GPU DB

CPU DB
GPU Database Roadmap

- **GPU memory capacity**: (8-80 GB)
- **CPU memory capacity**: (100GB-10TB)

**Performance**

**Data size**

- **GPU DB**
- **CPU DB**
GPU Database Roadmap

Project 1: Saturate GPU memory

- Saturate GPU memory bandwidth when data fits in GPU memory

**GPU Database Roadmap**

Project 1: **Saturate GPU memory** \(^1\)

Project 2: **Data compression** \(^2\)
  - More data can fit in GPU memory

---


GPU Database Roadmap

Project 1: **Saturate GPU memory** [1]
Project 2: **Data compression** [2]
Project 3: **Hybrid CPU-GPU DB** [3]

– Leverage both CPU and GPU computation

---

Outline

Project 1: Crystal library for in-GPU data analytics
Project 2: Data compression
Project 3: Hybrid CPU-GPU DB
Outline

**Project 1: Crystal library for in-GPU data analytics**

**Project 2: Data compression**

**Project 3: Hybrid CPU-GPU DB**
Issues with Prior Work on GPU Database

Past work reported wide variety of gains from $2 \times$ to $1000 \times$

One would expect the maximum gain to be roughly equal to the ratio of the memory bandwidth of GPU to that of CPU

**Key contribution**: developed Crystal library that allows GPU data analytics to saturate GPU memory bandwidth (V100, 880GB/s)
Shared memory (~100 KB) is local to every Streaming Multiprocessors (SM). Shared memory access is $10x$ faster than global memory.
Key Idea: Tile-Based Execution Model

Store intermediate result between operators in the shared memory (~10x faster).

(a) Conventional execution model
Key Idea: Tile-Based Execution Model

(a) Conventional execution model
Key Idea: Tile-Based Execution Model

(a) Conventional execution model
Key Idea: Tile-Based Execution Model

(a) Conventional execution model
Key Idea: Tile-Based Execution Model

Store intermediate result between operators in the shared memory (~10x faster)
Data is partitioned and processed in tiles (each tile must fit in shared memory)

(a) Conventional execution model

(b) Tile-based execution model
Key Idea: Tile-Based Execution Model

Store intermediate result between operators in the shared memory (~10x faster)
Data is partitioned and processed in tiles (each tile must fit in shared memory)

(a) Conventional execution model
(b) Tile-based execution model
Key Idea: Tile-Based Execution Model

Store intermediate result between operators in the shared memory (~10x faster)
Data is partitioned and processed in tiles (each tile must fit in shared memory)

(a) Conventional execution model
(b) Tile-based execution model
Experimental Results

With Crystal, GPU is on average **25X** faster than CPU running Star-Schema Benchmark (SSB)

GPU database with frequent PCIe data transfer can underperform a highly optimized CPU database
Outline

Project 1: Crystal library for in-GPU data analytics

**Project 2: Data compression**

Project 3: Hybrid CPU-GPU DB
Supported compression schemes: (1) frame-of-reference + bit-packing, (2) delta encoding, (3) run-length encoding

**Key challenge**: decompression should not be a performance bottleneck
Idea 1: **Tile-based decompression**

- Multiple decompression steps can be encapsulated into a single device function
- Decompression can be done inline with query execution

---

(a) Conventional decompression model

(b) Tile-based decompression model
GPU Data Compression — Key Ideas

### Idea 2: Efficient bit-packing compression

- Compact data format
- Low-level performance optimizations

---

**Data:**

- Block Size
- Miniblock Count
- Total Count
- Header
- (total count/block size) Blocks

**Block Starts:**

**Values:**

<table>
<thead>
<tr>
<th>100</th>
<th>101</th>
<th>101</th>
<th>102</th>
<th>101</th>
<th>101</th>
<th>102</th>
<th>101</th>
<th>99</th>
<th>100</th>
<th>105</th>
<th>107</th>
<th>114</th>
<th>112</th>
<th>110</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>101</td>
<td>101</td>
<td>102</td>
<td>101</td>
<td>101</td>
<td>102</td>
<td>101</td>
<td>99</td>
<td>100</td>
<td>105</td>
<td>107</td>
<td>114</td>
<td>112</td>
<td>110</td>
<td>105</td>
</tr>
</tbody>
</table>

**Reference miniblock**

- 99
- 1 2 2 3 2 2 3 2

**Miniblock**

- maxbits = 2
- 01 10 10 11 10 10 11 01

**Packing miniblock**

- maxbits = 4
- 0000 0001 0110 1000 1111 1101 1011 0110

**Encoded Block:**

- 99 2 4 01101011010110 00000001011010001111110110110110
Evaluation – Star Schema Benchmark

(a) Compressed data size

- Our compression rate (GPU-*) is comparable to the best-previous scheme (i.e. nvCOMP).
Evaluation – Star Schema Benchmark

- Our compression rate \((\text{GPU-\#})\) is comparable to the best-previous scheme (i.e. nvCOMP).
- \text{GPU-\#} is \textbf{2.2x} faster in decompression time than the best-previous scheme.
Our compression rate (GPU-*) is comparable to the best-previous scheme (i.e. nvCOMP).

GPU-∗ is 2.2x faster in decompression time than the best-previous scheme.

GPU-∗ is 2.6x faster in query running time than the best-previous scheme.

GPU-∗ executes queries with minimal performance degradation compared to no encoding.
Outline

Project 1: Crystal library for in-GPU data analytics
Project 2: Data compression

Project 3: Hybrid CPU-GPU DB
Heterogeneous GPU-CPU Data Analytics

Challenges in heterogeneous CPU-GPU data analytics

- **Data placement**: hot data should be cached in GPU memory
- **Heterogeneous query execution**: hybrid execution with minimal inter-device data transfer
GPU data caching should be:

1. **Fine-grained**: segment instead of column
2. **Semantic-aware**: priority of caching depends on the query pattern
Fine-grained Semantic-Aware Caching

Algorithm 1: Update the weighted frequency counter for segment $S$

1. $\text{UpdateWeightedFreqCounter}(\text{segment } S)$
   
   \begin{align*}
   &\# \text{ estimate query runtime when } S \text{ is not cached.} \\
   &RT_{\text{uncached}} = \text{estimateQueryRuntime}(\text{cached\_segments} \setminus S) \\
   &\# \text{ estimate query runtime when } S \text{ and segments correlated with } S \text{ are cached.} \\
   &RT_{\text{cached}} = \text{estimateQueryRuntime}(\text{cached\_segments} \cup S \cup \text{correlated\_segments}) \\
   &\text{weight} = RT_{\text{uncached}} - RT_{\text{cached}} \\
   &S.\text{weighted\_freq\_counter} += \text{weight} \\
   \text{for } C \text{ in correlated\_segments do} \\
   &\# \text{ evenly distribute weight to all segments correlated with } S \\
   &C.\text{weighted\_freq\_counter} += \text{weight} / |\text{correlated\_segments}| 
   \end{align*}

Weighted LFU replacement

- Each segment is assigned a different weight (higher weight $\rightarrow$ higher priority)

The weight of a segment reflects:

1. The relative speedup of caching a segment.
2. Correlation among segments from different columns

$\text{estimateQueryRuntime}()$ uses a model to predict runtime, assuming bandwidth as the bottleneck
Heterogeneous Query Execution

**Challenge:**
- Extra complexity of query execution due to only subset of data cached in GPU
- Query executor should fully exploit the data in GPU and coordinate query execution across two devices

**Solution:**
- Segment-level query execution
Segment level execution

- Group segments with the same execution plan into **segment groups**
Segment level execution

- Group segments with the same execution plan into **segment groups**
- Execute each segment group and merge the results

**SELECT S.D, SUM(R.C) FROM R, S**
WHERE R.B = S.D AND R.A > 10 AND S.E > 20
GROUP BY S.E
Heterogeneous Query Execution

Other optimizations

- **Late materialization**: express intermediate relation in the form of row ID to reduce the total data transfer
- **Operator pipelining**: pipelining consecutive operators on the same device whenever possible
- **Segment skipping**: apply minmax pruning both for predicate evaluation and join operator
Mordred Architecture

Three components:
- Cache Manager
- Query Optimizer
- Query Execution Engine
Evaluation — Caching Policies

Semantic-aware fine-grained caching achieves the best performance
Evaluation — SSB Runtime

Mordred (our design) is 6× faster than the best-previous design
– 8GB cache size, ~32GB data size
Conclusion

GPU database has great performance potential

Key challenge: large data sets do not fit in GPU memory

- Project 1: Saturate GPU memory
- Project 2: Data compression
- Project 3: Hybrid CPU-GPU DB

Future Directions

Performance

Data size

GPU memory capacity (8-80 GB)

CPU memory capacity (100GB-10TB)

[1] Anil Shanbhag, Samuel Madden, Xiangyao Yu, A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics, SIGMOD 2020
[3] Bobbi Yogatama, Weiwei Gong, Xiangyao Yu, Orchestrating Data Placement and Query Execution in Heterogeneous CPU-GPU DBMS, VLDB 2022
Special hardware designed for SQL analytics?
Transfer between CPU and GPU becomes a bottleneck?
How popular are GPUs used in industrial databases? What are the main barriers?
Next Lecture

DAWN workshop
- Reserve a presentation slot using the following google sheet
  https://docs.google.com/spreadsheets/d/1Re1M9FmJwl_YkidhNgeV0iKn-clssFrK_J1PMidaAuw/edit?usp=sharing
- 8-min per group (presentation + QA)

Project report (DDL: Dec. 19)
- **Submit to the hotcrp website** (like the proposal)

Submit course evaluation on [aefis.wisc.edu](http://aefis.wisc.edu)