Reproduction, Verification, and Improvements for SABER

Samuel Stark

29/11/2021

University of Cambridge

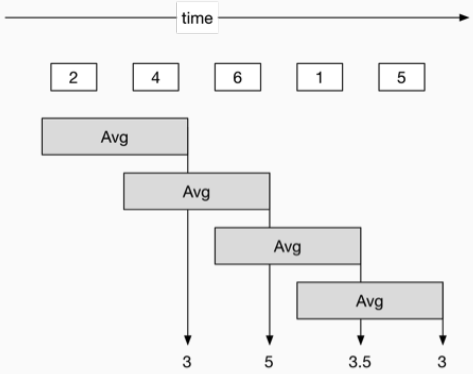
SABER

- Window-Based Hybrid Stream Processing
  - Executes Window-Based Streaming SQL queries

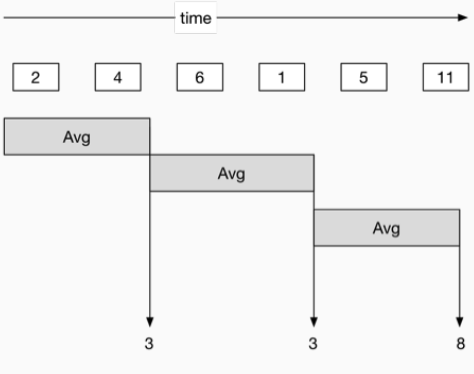
- Supports CPU and GPU
- Assign each task to the heterogenous processor that did it best last time
  - *Heterogenous Lookahead Scheduling*
  - Best = Highest Throughput

- Followed-up in 2020 with LightSaber
  - Focuses on multi-core processors
  - No GPU ;(

Stream Processing 101



Sliding Windows



Batch Windows

Images by Srinath Perera

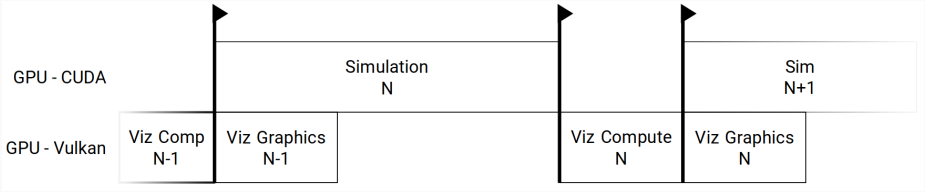
(Stream Processing 101: A Deep Look at Operators)

My Project

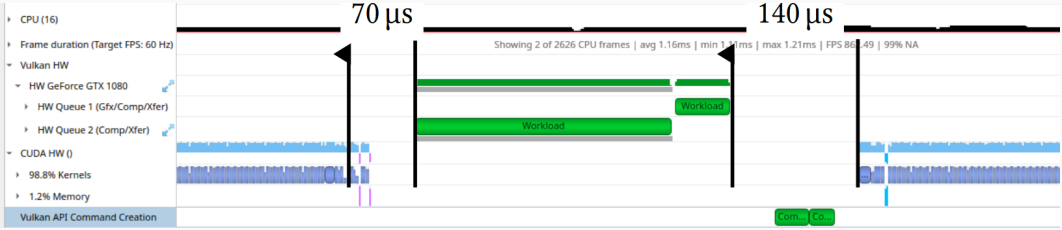
Reproduction

- First step - Verify my environment works by reproducing results
- Results have been replicated by ACM before, so this should be possible
- But Saber is from 2016, used older versions of software + OS
  - Ubuntu 14.04 (now 21.10)
  - gcc 4.8 (now 11.2!)

Verification - GPUs are finnicky



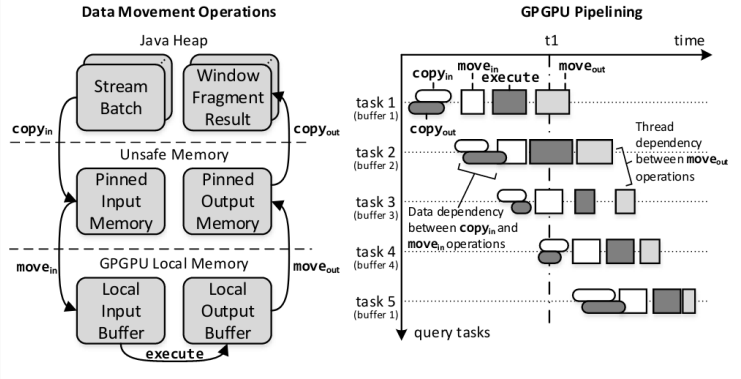
Expectation



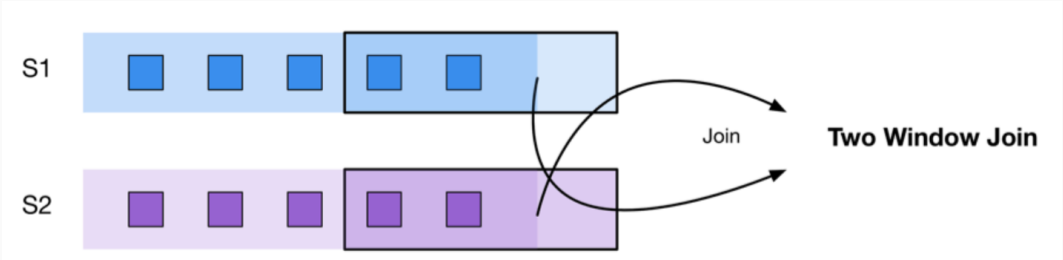
Reality

Verification

- The SABER paper explicitly plans out GPU usage, tries to avoid bubbles
- Use GPU profiling tools to check how well it actually does that



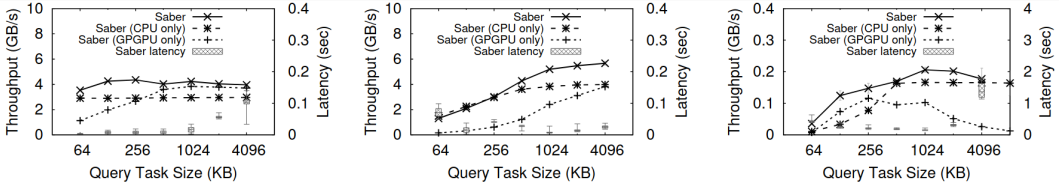
Stream Processing 102



Images by Srinath Perera
(Stream Processing 101: A Deep Look at Operators)

Improvements

- Theta-Join performance on GPU worsens as the task size increases
- Bottlenecked by the CPU generating window indices



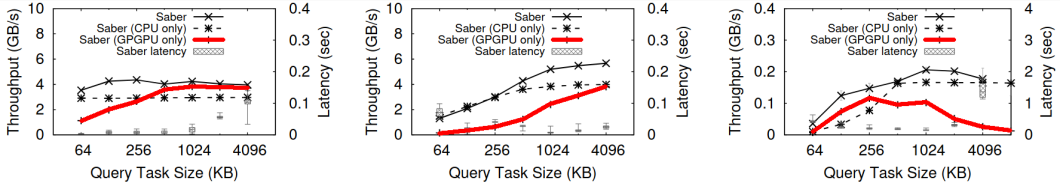
(a) SELECT_{10} with $\omega_{32\text{KB},32\text{KB}}$ **(b)** $\text{AGG}_{\text{avg}}\text{GROUP-BY}_{64}$ with $\omega_{32\text{KB},32\text{KB}}$ **(c)** JOIN_{4} with $\omega_{32\text{KB},32\text{KB}}$

Figure 12: Performance impact of query task size ϕ for different query types

"This is due to a limitation of our current implementation: the computation of the window boundaries is always executed on the CPU."

Improvements

- Theta-Join performance on GPU worsens as the task size increases
- Bottlenecked by the CPU generating window indices



(a) SELECT$_{10}$ with $\omega_{32\,\mathrm{KB},32\,\mathrm{KB}}$ **(b)** AGG$_{\mathrm{avg}}$GROUP-BY$_{64}$ with $\omega_{32\,\mathrm{KB},32\,\mathrm{KB}}$ **(c)** JOIN$_4$ with $\omega_{32\,\mathrm{KB},32\,\mathrm{KB}}$

Figure 12: Performance impact of query task size ϕ for different query types

"This is due to a limitation of our current implementation: the computation of the window boundaries is always executed on the CPU."

Improvements - Code

```
public void processData(...) {
   IQueryBuffer inputBuffer = batch.getBuffer();
   int start = batch.getBufferStartPointer();
   int end = batch.getBufferEndPointer();

   TheGPU.getInstance().setInputBuffer(qid, 0, inputBuffer, start, end);

   // ...
}
```

SelectionKernel.java

Improvements - Code

```
// CPU Code
public void processData(...) {
   // ...

   TheGPU.getInstance().setInputBuffer(qid, 0, inputBuffer1, start1, end1);
   TheGPU.getInstance().setInputBuffer(qid, 1, inputBuffer2, start2, end2);

   clearPointers ();
   computePointers (first, second);
   normalisePointers (start1);

   TheGPU.getInstance().setInputBuffer (qid, 2, startPointers);
   TheGPU.getInstance().setInputBuffer (qid, 3, endPointers);
   // ...
}
```

ThetaJoinKernel.java

Improvements - Code

```
// CPU Code
public void processData(...) {
   // ...
   clearPointers ();
   computePointers (first, second);
   normalisePointers (start1);
   // ...
}

private void computePointers(...) {
   while (currentIndex1 < endIndex1
         || currentIndex2 < endIndex2) {
      // ...
   }
}
```

ThetaJoinKernel.java

```
804  __kernel void computePointersKernel (
805          const int tuples,
806          const int inputBytes,
807          const int outputBytes,
808          const int _table_,
809          const int maxWindows,
810          const long previousPaneId,
811          const long batchOffset,
812          __global const uchar* input,
813          __global int* window_ptrs_,
814          __global int* _window_ptrs,
```

GPU Aggregation.cl
(Single-window version)

Summary

Reproduce

- Reproduce results on my hardware
- Examine any differences from the paper

Verify

- Run SABER under a GPU profiler
- Examine potential inefficiencies

Improve

- Try to port Join window-calculations to GPU
- Window-calc kernels already exist for other operations
- Examine results, or at least propose improvements

Summary

Reproduce

- Reproduce results on my hardware
- Examine any differences from the paper

Verify

- Run SABER under a GPU profiler
- Examine potential inefficiencies

Improve

- Try to port Join window-calculations to GPU
- Window-calc kernels already exist for other operations
- Examine results, or at least propose improvements

Summary

Reproduce

- Reproduce results on my hardware
- Examine any differences from the paper

Verify

- Run SABER under a GPU profiler
- Examine potential inefficiencies

Improve

- Try to port Join window-calculations to GPU
- Window-calc kernels already exist for other operations
- Examine results, or at least propose improvements

Summary

Reproduce

- Reproduce results on my hardware
- Examine any differences from the paper

Verify

- Run SABER under a GPU profiler
- Examine potential inefficiencies

Improve

- Try to port Join window-calculations to GPU
- Window-calc kernels already exist for other operations
- Examine results, or at least propose improvements

Summary

Reproduce

- Reproduce results on my hardware
- Examine any differences from the paper

Verify

- Run SABER under a GPU profiler
- Examine potential inefficiencies

Improve

- Try to port Join window-calculations to GPU
- Window-calc kernels already exist for other operations
- Examine results, or at least propose improvements

Questions?