Skipping Wavefronts in Pairwise Alignment

Alireza Mohammadidoost
PhD Student
UC San Diego

Supervisor: Prof. Yatish Turakhia
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Pairwise Sequence Alignment

- A way to find similarities between sequences

- Widely used in Bioinformatics for DNA and protein sequences

- The example above is global sequence alignment which is the main focus.
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Global Sequence Alignment

- It’s a 2-step algorithm using dynamic programming score matrix

<table>
<thead>
<tr>
<th></th>
<th>*</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Global Sequence Alignment

- It's a 2-step algorithm using dynamic programming score matrix

1. Score matrix calculation
   - Using the equation from top-left to bottom-right cell

\[
S(i, j) = \max \begin{cases} 
S(i - 1, j) + \text{gap} \\
S(i, j - 1) + (\text{mis})\text{match} \\
S(i, j - 1) + \text{gap} 
\end{cases}
\]
Global Sequence Alignment

- It's a 2-step algorithm using dynamic programming score matrix

1. Score matrix calculation
   - Using the equation from top-left to bottom-right cell
     \[
     S(i, j) = \max \begin{cases} 
     S(i - 1, j) + \text{gap} \\
     S(i - 1, j - 1) + \text{(mis)match} \\
     S(i, j - 1) + \text{gap} 
     \end{cases}
     \]
   - Storing the survivor path at every cell Memory of size $O(N^2)$ for trace-back information

<table>
<thead>
<tr>
<th></th>
<th>*</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Global Sequence Alignment

- It’s a 2-step algorithm using dynamic programming score matrix

1. Score matrix calculation
   - Using the equation from top-left to bottom-right cell
     \[
     S(i, j) = \max \begin{cases} 
     S(i-1, j) + \text{gap} \\
     S(i-1, j-1) + (\text{mis})\text{match} \\
     S(i, j-1) + \text{gap} 
     \end{cases}
     \]
   - Storing the survivor path at every cell Memory of size $O(N^2)$ for trace-back information

<table>
<thead>
<tr>
<th></th>
<th>*</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Global Sequence Alignment

- It’s a 2-step algorithm using dynamic programming score matrix

1. Score matrix calculation
   - Using the equation from top-left to bottom-right cell
     \[
     S(i, j) = \max\left\{ S(i-1, j) + \text{gap} , \ S(i-1, j-1) + (\text{mis})\text{match} , \ S(i, j-1) + \text{gap} \right\}
     \]
   - Storing the survivor path at every cell Memory of size $O(N^2)$ for trace-back information

<table>
<thead>
<tr>
<th></th>
<th>*</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G</strong></td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>-3</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>T</strong></td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Global Sequence Alignment

- It’s a 2-step algorithm using dynamic programming score matrix

1. Score matrix calculation
   - Using the equation from top-left to bottom-right cell
     \[
     S(i, j) = \max \begin{cases} 
     S(i - 1, j) + \text{gap} \\
     S(i - 1, j - 1) + (\text{mis})\text{match} \\
     S(i, j - 1) + \text{gap} 
     \end{cases}
     \]
   - Storing the survivor path at every cell Memory of size $O(N^2)$ for trace-back information

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
& * & A & G & C & A & T \\
\hline
* & 0 & -1 & -2 & -3 & -4 & -5 \\
C & -1 & 0 & -1 & -1 & -2 & -3 \\
G & -2 & -1 & 1 & 0 & -1 & -2 \\
A & -3 & -1 & 0 & 1 & 1 & 0 \\
T & -4 & -2 & -1 & 0 & 1 & 2 \\
A & -5 & -3 & -2 & -1 & 1 & 1 \\
\hline
\end{array}
\]
Global Sequence Alignment

- It’s a 2-step algorithm using dynamic programming score matrix

2. Path trace back
  - Tracing back the survivor path from bottom-right to top-left to find the alignment

AGCAT_
CGATA
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Motivation

- Pairwise sequence alignment is one of the most costly kernels in all of bioinformatics
  - Dominates most of the runtime of many applications, can be up to 80%

- **Wavefront parallelism** is one approach to accelerate DP matrix alignment by exploiting parallelism of anti-diagonal independent elements
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Prior Works on Hardware acceleration

- LOGAN is a Multi-GPU implementation of pairwise sequence alignment
- Exploits wavefront parallelism by computing anti-diagonal cells in parallel
- 3 wavefront buffers in global memory to avoid shared memory bottleneck in long sequences

(Zeni et al., IPDPS 2020)
Prior Works on Hardware acceleration

- Wavefronts are divided into segments and threads calculate each segment in an iteration of a loop.
  - Not restricted by sequence length
- LOGAN calculates final score of the alignment for long sequences with high performance
- LOGAN ignores trace back step and doesn’t find the actual alignment.

(Zeni et al., IPDPS 2020)
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront

![Diagram](image-url)
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront

- Skipping some wavefronts while keeping 2 consecutive ones at each step
  - No need to synchronization for wavefronts not written
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront

- Skipping some wavefronts while keeping 2 consecutive ones at each step
  - No need to synchronization for wavefronts not written
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront

- Skipping some wavefronts while keeping 2 consecutive ones at each step
  - No need to synchronization for wavefronts not written
  - More complex dependency
Wavefront Skipping Approach

- The same wavefront parallelism approach
  - Synchronization after writing each wavefront

- Skipping some wavefronts while keeping 2 consecutive ones at each step
  - No need to synchronization for wavefronts not written
  - More complex dependency

- Complexity increases with larger number of skipped wavefronts (K)
Wavefront Skipping Approach

- Avoiding the complexity by storing skipped wavefronts in shared memory
- Converting some of GM synchronizations to SM synchronizations which are less expensive
- Restriction to sequence length due to the limited size of the shared memory
Wavefront Skipping Approach

- The segmentation approach used in LOGAN can be employed
Wavefront Skipping Approach

- The segmentation approach used in LOGAN can be employed
- The results can be overwritten to the same buffers.
  - Less buffers than LOGAN needed
- The last column should be stored
Wavefront Skipping Approach

- The segmentation approach used in LOGAN can be employed

- The results can be overwritten to the same buffers.
  - Less buffers than LOGAN needed

- The last column should be stored
Wavefront Skipping Approach

- The segmentation approach used in LOGAN can be employed.
- The results can be overwritten to the same buffers.
  - Less buffers than LOGAN needed.
- The last column should be stored.
- Shared memory depends on segment length.
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Benefits of Wavefront Skipping in Global Alignment

- Reduction in the number of GM synchronizations and replacing them with SM synchronizations
- Reduction in global memory access
- Less buffers needed to be allocated in global memory
Overview

- Pairwise Sequence Alignment
- Global Sequence Alignment
- Motivation
- Prior Works on Hardware acceleration
- Wavefront Skipping Approach
- Benefits of Wavefront Skipping in Global Alignment
- Trace back Memory Mapping
Trace back Memory Mapping

- In order to have coalescing access, the pointers should be indexed consecutively in each wavefront
Trace back Memory Mapping

- In order to have coalescing access, the pointers should be indexed consecutively in each wavefront
Trace back Memory Mapping

- In order to have coalescing access, the pointers should be indexed consecutively in each wavefront
Trace-back Memory Mapping

- In order to have coalescing access, the pointers should be indexed consecutively in each wavefront.
Trace back Memory Mapping

- In order to have coalescing access, the pointers should be indexed consecutively in each wavefront.

- Each pointer is stored in 2 bits. Therefore, each byte has 4 pointers.
Trace-back Memory Mapping

- In order to have coalescing access, the pointers should be indexed consecutively in each wavefront.

- Each pointer is stored in 2 bits. Therefore, each byte has 4 pointers.

- Assuming \( I \) as the pointer index, it can be stored at the following address:
  \[
  \left( \frac{I}{4} \right) \cdot (I \% 4)
  \]
  - Represented as (byte address).(byte offset from 0 to 3)
Trace back Memory Mapping

- Example

  Threads# = 4

  Trace back matrix size = 30×30 ÷ 4 = 225

  Wavefront Index = 24

  Strat index = 304

  Mapping function: \( \left( \frac{I}{4} \right) \cdot (I \% 4) \)
Trace back Memory Mapping

- Example

  Threads\# = 4

  Trace back matrix size = 30×30 ÷ 4 = 225

  Wavefront Index = 24

  Strat index = 304

  Mapping function: \( \left( \frac{I}{4} \right) \cdot (I\%4) \)
Trace back Memory Mapping

- Example

  Threads\# = 4
  
  Trace back matrix size = 30\times30 \div 4 = 225
  
  Wavefront Index = 24
  
  Strat index = 304

  Mapping function: \( \left( \frac{I}{4} \right) \cdot (I \% 4) \)
Trace back Memory Mapping

- Example

  $\text{Threads}\# = 4$
  
  $\text{Trace back matrix size} = 30 \times 30 \div 4 = 225$
  
  $\text{Wavefront Index} = 24$
  
  $\text{Strat index} = 304$

  $\text{Mapping function:} \left( \frac{I}{4} \right) \cdot (I \% 4)$
Trace back Memory Mapping

- Example

  Threads\# = 4
  
  Trace back matrix size = 30×30 ÷ 4 = 225
  
  Wavefront Index = 24
  
  Strat index = 304
  
  Mapping function: \( \left( \frac{I}{4} \right) \cdot (I \% 4) \)
Trace back Memory Mapping

- Example

  Threads\# = 4

  Trace back matrix size = 30\times 30 \div 4 = 225

  Wavefront Index = 24

  Strat index = 304

  Mapping function: \left(\frac{I}{4}\right) \cdot (I\%4)

- Threads want to access the same byte that implies using atomic instructions
Trace back Memory Mapping

- Instead, we can use another mapping

\[(I\%MS) \cdot \left(\frac{I}{MS}\right)\]

MS: Matrix size

<table>
<thead>
<tr>
<th>Memory Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>\vdots</td>
</tr>
<tr>
<td>MS-1</td>
</tr>
</tbody>
</table>
Trace back Memory Mapping

- Instead, we can use another mapping

\[(I \% MS). \left( \frac{I}{MS} \right)\]

MS: Matrix size
Trace back Memory Mapping

- Instead, we can use another mapping

\[(I \% MS) \cdot \left( \frac{I}{MS} \right)\]

MS: Matrix size
Trace back Memory Mapping

- Instead, we can use another mapping

\[(I\%MS).\left(\frac{I}{MS}\right)\]

MS: Matrix size
Trace back Memory Mapping

- Instead, we can use another mapping

\[(I \% MS). \left( I \over MS \right) \]

MS: Matrix size
Trace back Memory Mapping

- Example

Threads\# = 4

Trace back matrix size = 30×30 ÷ 4 = 225
Wavefront Index = 24
Strat index = 304

Mapping function: \((I\%MS).\left(\frac{I}{MS}\right)\)
Trace back Memory Mapping

- Example

  Threads\# = 4
  Trace back matrix size = 30×30 \div 4 = 225
  Wavefront Index = 24
  Strat index = 304

  Mapping function: \( I\%MS \cdot \left( \frac{I}{MS} \right) \)

- Using this mapping function for different sequence lengths, speedup of 2.5x can be gained for the entire alignment.
Conclusion

- Pairwise sequence alignment is a basic building-block in Bioinformatics.
- GPU implementation of this algorithm is based on wavefront parallelism.
- Skipping some wavefronts can be beneficial.
- Most of the works done on GPU haven’t implemented the trace-back step.
- Using a proper mapping function, trace-back can be done efficiently.
Thank you!