Filter Before You Parse
Accelerating Any Sequence Aligner

Mohammed Alser, PhD
@mealser

RECOMB 2023 - Bio-Arch
14 April 2023
Pairwise Sequence Alignment

- Calculating the best arrangement of matches and edits (i.e., insertions, deletions, or substitutions) needed to make one sequence exactly matches the other one.

For 30K long sequence pairs, the DP table size = 900MB

For 300K long sequence pairs, the DP table size = 90GB
Partial Computation of DP Table

- Different algorithms compute the DP table differently.

Koerkamp+, "Exact global alignment using A* with seed heuristic and match pruning", bioRxiv, 2022
Alignment is Major Bottleneck

>60% of the read mapper’s execution time is spent in sequence alignment

ONT FASTQ size: 103MB (151 reads), Mean length: 356,403 bp, std: 173,168 bp, longest length: 817,917 bp
Goal: Minimizing Alignment Time

Sequence Alignment is expensive

Goal: Accelerate ANY sequence aligner by reducing the need for dynamic programming algorithms
Key Idea

Genomic Strings

- Dissimilar Strings: Ignore them if the number of differences exceeds a threshold.
- Similar Strings: Find number, location, and type of differences?

EXPENSIVE!
1. **Filter out** most of incorrect mappings.
2. **Preserve** all correct mappings.
3. **Do it quickly.**
How It Works?
Mohammed Alser, Taha Shahroodi, Juan-Gomez Luna, Can Alkan, and Onur Mutlu, "SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs" Bioinformatics, 2020.
[Source Code]
[Online link at Bioinformatics Journal]
Key observation:

- Correct alignment is a sequence of non-overlapping long matches.

*Dot plot, dot matrix* (Lipman and Pearson, 1985)

*T. ophioglossoides* scaffolds

*T. infatum* scaffolds
**SneakySnake**

- **Key observation:**
  - Correct alignment is a sequence of non-overlapping long matches

- **Key idea:**
  - Approximate edit distance calculation is similar to Single Net Routing problem in VLSI chip
SneakySnake Walkthrough

Building Neighborhood Map  Finding the Optimal Routing Path  Examining the Snake Survival

Given two genomic sequences, a reference sequence \( R[1 \ldots m] \) and a query sequence \( Q[1 \ldots m] \), and an edit distance threshold \( E \), we calculate the entry \( Z[i,j] \) of the chip maze, where \( 1 \leq i \leq (2E + 1) \) and \( 1 \leq j \leq m \), as follows:

\[
Z[i,j] = \begin{cases} 
0, & \text{if } i = E + 1, \ Q[j] = R[j], \\
0, & \text{if } 1 \leq i \leq E, \ Q[j-i] = R[j], \\
0, & \text{if } i > E + 1, \ Q[j+i-E-1] = R[j], \\
1, & \text{otherwise} 
\end{cases}
\]  

(1)

\[
E = 3
\]

<table>
<thead>
<tr>
<th>column</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>3\text{rd} Upper Diagonal</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2\text{nd} Upper Diagonal</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1\text{st} Upper Diagonal</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Main Diagonal</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1\text{st} Lower Diagonal</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2\text{nd} Lower Diagonal</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3\text{rd} Lower Diagonal</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
# SneakySnake Walkthrough

## Building Neighborhood Map

<table>
<thead>
<tr>
<th>Column</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd Upper Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2nd Upper Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st Upper Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st Lower Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2nd Lower Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Lower Diagonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Finding the Optimal Routing Path

- **Entrance**: 3rd Upper Diagonal
- **Exit**: 3rd Lower Diagonal
- **E = 3**

---

**SAFARI**
SneakySnake Walkthrough

Building Neighborhood Map  Finding the Optimal Routing Path  Examining the Snake Survival

Entrance  Exit

SAFARI
This is what you actually need to **build** and it can be done **on-the-fly**!
Evaluation Results
Fig. 11: The execution time of SneakySnake, KSW2, and SneakySnake integrated with KSW2 using long sequences, (a) 10Kbp and (b) 100Kbp, and a single CPU thread. The left y-axes of (a) and (b) are on a logarithmic scale. For each edit distance threshold value, we provide in the right y-axes of (a) and (b) the rate of accepted pairs (out of 100,000 pairs for 10Kbp and out of 74,687 pairs for 100Kbp) by SneakySnake that are passed to KSW2. We present the end-to-end speedup values obtained by integrating SneakySnake with KSW2.
Accelerating WFA (and BiWFA)

- Integrating SneakySnake with WFA accelerates end-to-end sequence alignment by about 4.2-9.6x for different sequence lengths.

![Speedup bar chart]

- WFA
- SS+WFA

Sequence lengths

- 100
- 250
- 1000
- 10000

Speedup

0 1 2 3 4 5 6 7 8 9 10

Safari
Key Results of SneakySnake

- SneakySnake is up to four orders of magnitude more accurate than Shouji (Bioinformatics’19) and GateKeeper (Bioinformatics’17)

- Using short reads, SneakySnake accelerates Edlib (Bioinformatics’17) and Parasail (BMC Bioinformatics’16) by
  - up to 37.7× and 43.9× (>12× on average), on CPUs
  - up to 413× and 689× (>400× on average) with FPGA/GPU acceleration

- Using long reads, SneakySnake accelerates Parasail and KSW2 by 140.1× and 17.1× on average, respectively, on CPUs
Can We Do Better?

Alleviating Data Movement Bottlenecks
Near-memory Pre-alignment Filtering

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gomez-Luna, Henk Corporaal, Onur Mutlu,

“FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications“


[Source Code]
Heterogeneous System: CPU+FPGA

We evaluate two POWER9+FPGA systems:

1. HBM-based AD9H7 board: Xilinx Virtex Ultrascale+™ XCVU37P-2
2. DDR4-based AD9V3 board: Xilinx Virtex Ultrascale+™ XCVU3P-2

FPGA + HBM on the same package substrate
Near-memory pre-alignment filtering improves performance and energy efficiency by 27.4× and 133×, respectively, over a 16-core (64 hardware threads) IBM POWER9 CPU.
GRIM-Filter

Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu,
"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"
Proceedings of the 16th Asia Pacific Bioinformatics Conference (APBC), Yokohama, Japan, January 2018.
arxiv.org Version (pdf)

BMC Genomics
Research | Open Access | Published: 09 May 2018

GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan & Onur Mutlu

BMC Genomics 19, Article number: 89 (2018) | Cite this article
4340 Accesses | 39 Citations | 9 Altmetric | Metrics
GRIM-Filter in 3D-Stacked DRAM

- Each DRAM layer is organized as an array of banks
  - A bank is an array of cells with a row buffer to transfer data
GRIM-Filter: Bitvectors

- Represent each bin with a bitvector that holds the occurrence of all permutations of a small string (token) in the bin.

- To account for matches that straddle bins, we employ overlapping bins.
  - A read will now always completely fall within a single bin.
GRIM-Filter: Bitvectors

Storing all bitvectors requires $4^n \times t$ bits in memory, where $t =$ number of bins & $n =$ token length.

For bin size $\sim$200, and $n = 5$, memory footprint $\sim 3.8$ GB.
GRIM-Filter in 3D-Stacked DRAM

- Each DRAM layer is organized as an array of **banks**
  - A **bank** is an array of cells with a row buffer to transfer data

- The layout of bitvectors in a bank enables filtering many bins in parallel
**GRIM-Filter**

- **Key observation:** FPGA and GPU accelerators are Heavily bottlenecked by Data Movement.

- **Key idea:** exploiting the high memory bandwidth and the logic layer of 3D-stacked memory to perform highly-parallel filtering in the DRAM chip itself.

- **Key results:**
  - We propose an algorithm called GRIM-Filter
  - GRIM-Filter with processing-in-memory is 1.8x-3.7x (2.1x on average) faster than FastHASH filter (BMC Genomics’13) across real data sets.
  - GRIM-Filter has 5.6x-6.4x (6.0x on average) lower falsely accepted pairs than FastHASH filter (BMC Genomics’13) across real data sets.
Key Conclusion

Most speedup comes from parallelism enabled by novel architectures and algorithms
Filter Before You Parse
Accelerate Any Sequence Aligner

Mohammed Alser, PhD
@mealser

RECOMB 2023 - Bio-Arch
14 April 2023