Accelerating Genomic Sequence Alignment Workload with Scalable Vector Architecture

Dong-hyeon Park, Jon Beaumont, Trevor Mudge

University of Michigan, Ann Arbor
Genomics

Past

Weeks ~$3 billion

“Human Genome Project”, 2004

Present

~13 hours <$10,000

“SpeedSeq”, 2015

Future Applications

On-Demand Diagnosis

DNA Storage

Forensics

Portable Sequencer

Human Genome:

3.2 billion base pairs

Need to sample at 30-50x coverage
Whole Genome Sequencing Pipeline

10-20k in length

Read/Extract Sequences
- Reading fragment samples of whole genome
- Signal/Image processing

Sequence Alignment
- Matching overlaps across multiple sequences
- Dynamic vs heuristic algorithm

Assembly
- Reconstructing the original sequence
- de-novo vs mapping assembly

Analysis
- Identifying gene variants and abnormalities
- Pattern matching, HMM, DNN

Reference gene

Reconstructed sequence
Target Architecture:

Scalable Vector Extension (SVE)
ARM’s Scalable Vector Extension (SVE)

• Designed to complement existing SIMD architecture (NEON)

• Key Features:
  • Scalable Vector Length (128 - 2048-bits)
  • Per-lane Predication (32 SIMD Reg. + 16 Predicate Reg.)
  • Gather-load and scatter-store
  • Horizontal vector operations

Vector Length Agnostic Code
ARM’s Scalable Vector Extension (SVE)

• Genomic sequences are sampled at different lengths depending on the device used for sampling:
  • Illumina HiSeq System: 30-300 bps
  • Sanger 3730xl: 400-900 bps

Vector-Length Agnostic Code can be used to **Dynamically Choose the Optimal SIMD Width**
Target Algorithm:

Smith-Waterman
Sequence Alignment
Smith-Waterman Algorithm

Local sequence alignment algorithm developed in 1981

Inputs:
- Reference Sequence
- Query Sequence

Output:
- Alignment Location & Score

Smith-Waterman

Scoring Matrix Construction

Matrix Backtracking
### Scoring Matrix Construction

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**Scoring Matrix**

$$H(m, n) = \max \begin{cases} E(m, n) \\ F(m, n) \\ H(m - 1, n - 1) + S(a_m, b_n) \end{cases}$$

$$E(m, n) = \max \begin{cases} H(m, n - 1) - g_o \\ E(m, n - 1) - g_e \end{cases}$$

$$F(m, n) = \max \begin{cases} H(m - 1, n) - g_o \\ F(m - 1, n) - g_e \end{cases}$$
Scoring Matrix Construction

<table>
<thead>
<tr>
<th>Query Sequence</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scoring Matrix

\[
\begin{align*}
H(m, n) &= \max \left\{ E(m, n), F(m, n) \right\} \\
E(m, n) &= \max \left\{ H(m, n - 1) - g_o, E(m, n - 1) - g_e \right\} \\
F(m, n) &= \max \left\{ H(m - 1, n) - g_o, F(m - 1, n) - g_e \right\}
\end{align*}
\]
Backtracking

Finds the best local alignment from the scoring matrix

Step 2.
Check the adjacent entries for the next largest score
Move to the entry with the largest score and continue the path
**Backtracking**

Finds the best local alignment from the scoring matrix

**Step 3.**

Get the resulting alignment

<table>
<thead>
<tr>
<th>Path Direction</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>Deletion</td>
</tr>
<tr>
<td>Vertical</td>
<td>Insertion</td>
</tr>
<tr>
<td>Diagonal</td>
<td>Match</td>
</tr>
</tbody>
</table>

---

**Backtracking**

<table>
<thead>
<tr>
<th>Query Sequence</th>
<th>Reference Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

- **max entry**
- Traverse back through the largest score

---

**Step 2.**

Reference Sequence:
- A
- C
- A
- C
- A
- A

Query Sequence:
- A
- C
- A
- A
- C
- C

Matrix:
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Backtracking

Finds the best local alignment from the scoring matrix

Step 3.

Get the resulting alignment

Reference: A–CAC
Query: AGCAC
Insertion

Alignment Score: 7
**Vectorization**

**BATCH:**

Reference

Query

**SLICED:**

(striped)

**Wavefront:**

Alignment Location & Score


[4] Rognes
Batch Smith-Waterman

Reference Sequence

Query Sequences Sampled

SVE[0] Query 0 Reference 0
SVE[1] Query 1 Reference 1
SVE[K] Query K Reference K

[4] Rognes
Sliced Smith-Waterman
(striped)

Reference Sequence

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<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></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></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></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></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>

Query Sequence

Initial Calculation of $H(m,n)$

$V_L = K$

$SVE[0] \quad SVE[1] \quad ... \quad SVE[K-1]$

$$H(m, n) = \max \begin{cases} E(m, n) \\ F(m, n) \\ H(m-1, n-1) + S(a_m, b_n) \end{cases}$$

### Sliced Smith-Waterman (striped)

**Reference Sequence**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<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>A</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>G</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>C</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>A</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>...</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>

**Query Sequence**

- A
- G
- C
- A
- ...

**Value of F need to be re-calculated**

Horizontal dependencies between slices not accounted

---

VL = K

SVE[0] SVE[1] ... SVE[K-1]

Sliced Smith-Waterman (striped)

Reference Sequence

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<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></td>
</tr>
<tr>
<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></td>
</tr>
<tr>
<td>A</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>

Resolve Dependencies

2

\[ F(m, n) = \max \begin{cases} H(m - 1, n) - g_o \\ F(m - 1, n) - g_e \end{cases} \]

\[ H(m, n) = \max(F(m, n), H(m, n)) \]

Wavefront Smith-Waterman

Reference Sequence

<table>
<thead>
<tr>
<th>--</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>F, E</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></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>
<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>
<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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All dependency comes from previous execution
Wavefront Smith-Waterman

Reference Sequence

All dependency comes from previous execution

Wavefront Smith-Waterman

All dependency comes from previous execution

<table>
<thead>
<tr>
<th>Query Sequence</th>
<th>Reference Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A C A C A G T A C A</td>
</tr>
<tr>
<td>A</td>
<td>H F,E 3</td>
</tr>
<tr>
<td>A</td>
<td>H F,E 2</td>
</tr>
<tr>
<td>G</td>
<td>F,E 1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

Wavefront Smith-Waterman

Reference Sequence

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>F,E</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>F,E</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>F,E</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>F,E</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

More book-keeping overhead than other algorithms:
- Keep track of H values of two prev. iterations
- F and E values from prev. iteration

Experimental Evaluation:

Smith-Waterman on gem5 w/ SVE
Experimental Setup

Gem5 Simulator w/ ARM SVE Simulation

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Single-Core out-of-order 64-bit ARM, 1GHz, 8-issue SIMD Width: 128-bit (NEON), 128/256/512/1024-bit (SVE)</td>
</tr>
<tr>
<td>Cache</td>
<td>32KB private L1 instruction cache, 2-way associative 64KB private L1 data cache, 2-way associative 4MB private L2 inclusive cache, 8-way associative</td>
</tr>
<tr>
<td>DRAM</td>
<td>Capacity: 8GB Latency: 30 ns Memory Controller Bandwidth: 12.8 GB/s</td>
</tr>
</tbody>
</table>
Experimental Setup

Application:

Smith-Waterman – Batch, Sliced, and Wavefront

- Reference:
  25-400 bps samples from *E. Coli 536* Gene (4.9 Mbps)
- Query:
  1000 x 25-400 bps samples through WGSim
Advantage of SVE over Traditional System

- CPU, NEON implementation written in C. SVE hand-written in assembly.
- SVE outperforms both CPU and NEON implementations by at least 3x
- Batch, Sliced and Wavefront used 32-bit, 16-bit and 64-bit vectors respectively.

Alignment Time Speedup over Baseline CPU

<table>
<thead>
<tr>
<th>Sequence Length</th>
<th>cpu</th>
<th>batch_neon</th>
<th>sliced_neon</th>
<th>batch_sve_128bit</th>
<th>sliced_sve_128bit</th>
<th>wavefront_128bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1</td>
<td>1.5</td>
<td>3.3</td>
<td>9.7</td>
<td>1.5</td>
<td>2.6</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>1.5</td>
<td>0.4</td>
<td>9.7</td>
<td>0.8</td>
<td>1.4</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1.4</td>
<td>0.8</td>
<td>8.2</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>1.4</td>
<td>1.4</td>
<td>7.8</td>
<td>1.4</td>
<td>2.1</td>
</tr>
<tr>
<td>400</td>
<td>1</td>
<td>1.3</td>
<td>2.1</td>
<td>25</td>
<td>1.2</td>
<td>38.2</td>
</tr>
</tbody>
</table>

Speedup over CPU
Impact of Handwritten Assembly

- Hand-written assembly code of Wavefront Algorithm has 4-6x speedup over C code.

### Speedup of Wavefront Algorithm

<table>
<thead>
<tr>
<th>Sequence Length (bps)</th>
<th>Speedup of Wavefront Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>L=25</td>
<td>CPU (naïve) 1.88, CPU (wav) 3.82, CPU-ASSEM 6.15, SVE-128bit 5.77, SVE-256bit 8.64</td>
</tr>
<tr>
<td>L=50</td>
<td>CPU (naïve) 1.25, CPU (wav) 1.00, CPU-ASSEM 8.64, SVE-128bit 5.09, SVE-256bit 11.91</td>
</tr>
<tr>
<td>L=100</td>
<td>CPU (naïve) 1.59, CPU (wav) 1.00, CPU-ASSEM 8.98, SVE-128bit 5.09, SVE-256bit 14.34</td>
</tr>
<tr>
<td>L=200</td>
<td>CPU (naïve) 1.69, CPU (wav) 1.00, CPU-ASSEM 9.24, SVE-128bit 5.38, SVE-256bit 15.95</td>
</tr>
</tbody>
</table>
Advantage of SVE over Traditional System

- SVE reduces the instruction execution significantly compared to CPU or NEON
Memory Bandwidth Comparison

- Sliced and Wavefront significantly reduce the memory bandwidth compared to the Batch algorithm
Vector Scaling of Different Algorithms

- Batch and Sliced show marginal improvement with increasing vector length
- Difficult to keep up with increased memory demand
- Need to resolve dependencies.

Vector Performance Scaling of Batch, Sliced, and Wavefront at Sequence Length of L=100

- Batch and Sliced show marginal improvement with increasing vector length
- Difficult to keep up with increased memory demand
- Need to resolve dependencies.

Vector Performance Scaling of Batch, Sliced, and Wavefront at Sequence Length of L=400

- Batch and Sliced show marginal improvement with increasing vector length
- Difficult to keep up with increased memory demand
- Need to resolve dependencies.
Fixed HW Options: Batch vs Sliced vs Waveform @512-bit

Comparison of Different Smith-Waterman Vectorization
512-bit SVE and 64kB L1D Cache

Execution Time - Log scale (ms)

Batch
• Low overhead.
• Poor scaling.
• Fastest for short seq.

Waveform
• Efficient use of vector Lanes
• Fastest for medium seq.

Sliced
• High overhead.
• Execution bypassing
• Fastest for long seq.
HW with Variable Vector Length

- Given freedom to choose the hardware for each sequence length, we can establish a set of optimal algorithm-hardware pair.

<table>
<thead>
<tr>
<th>Read Length</th>
<th>Algorithm</th>
<th>Vector Length</th>
<th>Speedup Over 512-bit Wavefront</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 50 bps</td>
<td>Batch</td>
<td>128-bit</td>
<td>2.77</td>
</tr>
<tr>
<td>50-100 bps</td>
<td>Wavefront</td>
<td>1024-bit</td>
<td>1.03</td>
</tr>
<tr>
<td>100-400 bps</td>
<td>Sliced</td>
<td>256-bit</td>
<td>1.23-3.06</td>
</tr>
</tbody>
</table>
Conclusion

Smith-Waterman on SVE:

+ Select Optimal Vector Length & Algorithm depending on Input
+ Lower Instruction Footprint

- Improvements to memory controller can lead to improved performance

- Wavefront algorithm use 64-bit vectors due to limitations on gather-scatter instruction addressing.
Key References


Questions?