Genome-on-Diet
Taming Large-Scale Genomic Analyses via Sparsified Genomics

Mohammed Alser, PhD
@mealser

RECOMB 2023 – BIO-Arch
14 April 2023
Genome Sequencing is Rapidly Growing

Specialized Machine for Sequencing

- 35 Gb/30h
- 6 Tb/44h
- 14 Tb/72h
Genome Sequencing is Rapidly Growing

Specialized Machine for Sequencing
- 6 Tb/44h
- 14 Tb/72h
- 35 Gb/30h

General-Purpose Machine for Analysis
- 6 Tb/44h
Lack of Specialized Compute Capability

**Specialized Machine for Sequencing**
- 35 Gb/30h
- 6 Tb/44h

**General-Purpose Machine for Analysis**
- 14 Tb/72h

**FAST**

**SLOW**

SAFARI
Improving Processing via Accelerators

Specialized Genomic Accelerators (GPU, FPGA)

- Scrooge [Bioinformatics 2023]
- RUBICON [arXiv 2022]
- GateKeeper-GPU [IPDPSW 2021]
- SneakySnake [Bioinformatics'20]
- Shouji [Bioinformatics 2019]
- MAGNET [AACBB 2018]
- GateKeeper [Bioinformatics 2017]
Data movement is a major bottleneck in modern computer architectures.

Over 60% of the total system energy is spent on data movement.

A. Boroumand et al., “Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks,” ASPLOS, 2018
Improving processing via paradigm shift

Near-memory/In-memory Genomic Accelerators

- **AIM** [Bioinformatics 2023]
- **SeGraM** [ISCA 2022]

Specialized Genomic Accelerators (GPU, FPGA)

- **Scrooge** [Bioinformatics 2023]
- **RUBICON** [arXiv 2022]

Improving **performance and energy efficiency** by 1-3 orders of magnitude

In-storage Sequence Alignment

- **GenStore** [ASPLOS 2022]
Our Goal

To further reduce the execution time and memory/storage footprint of genomic analyses via **sparsified genomics**.
Sparsifying Genomic Data

ACCCTAAACCCTAACCCTAACCCTAACCCTAA

Exact Match

ACCCTAAACCCTAACCCTAACCCTAACCCTAA
Sparsifying Genomic Data

ACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAA

A_C_T_A_C_T_A_C_T_A_C_T_A_C_T_A_C_T_A

N Bytes

N/2 Bytes

Still Exact Match

ACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAA

A_C_T_A_C_T_A_C_T_A_C_T_A_C_T_A_C_T_A_A

SAFARI
Sparsifying Genomic Data Is Challenging

ACCCTAACCCTAACCCTAACCCTAACCCTAA

A_C_T_A_C_T_A_C_T_A_C_T_A_C_T_A
A_C_T_A_C_T_A_C_T_A_C_T_A_C_T_A
C_T_A_C_T_A_C_T_A
T_A_C_T_A_C_T_A
A_C_T_A_C_T_A_C_T_A

__

C_C_A_C_C_A_C_C_A_C_C_A_C_C_A_C
C_C_A_C_C_A_C_C_A_C_C_A_C_C_A_C
C_C_A_C_C_A_C_C_A_C_C_A_C_C_A_C
C_C_A_C_C_A_C_C_A_C_C_A_C_C_A_C
C_C_A_C_C_A_C_C_A_C_C_A_C_C_A_C

No Match 😞
How It Works?
Genome-on-Diet Steps

- Compressed Indexing
- Pattern Alignment
- Compressed Seeding
- Location Voting
- Sequence Alignment
Step 1: Compressed Indexing

- We use a user-defined binary pattern to identify the location and number of the to-be-dropped bases.

| Genome sequence: | A | C | C | C | T | A | G | C | C | C | T | A | A | G |
| Diet pattern:     | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Patterned genome: | A | C | T | G | C | T | A |

- Index (e.g., hash table)
Step 2: Pattern Alignment

- Deciding *where* in the read to apply the pattern *essential* for the correctness of Genome-on-Diet

<table>
<thead>
<tr>
<th>Read sequence:</th>
<th>G A C C C T A G C C C T A A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diet pattern 0:</td>
<td>1 0 1 0 1 0 1 0 1 0 1 0</td>
</tr>
<tr>
<td>Patterned read 0:</td>
<td>G C C A C C A A</td>
</tr>
<tr>
<td>k-mers:</td>
<td>G C C A</td>
</tr>
</tbody>
</table>

Don’t exist in the index

2 ≥ 0? thus the correct shift amount = 1

<table>
<thead>
<tr>
<th>Shift amount = 1</th>
<th>1 0 1 0 1 0 1 0 1 0 1 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patterned read 1:</td>
<td>G A C C C T A G C C C T A A</td>
</tr>
<tr>
<td>k-mers:</td>
<td>A C T G C T A</td>
</tr>
<tr>
<td>Exist in the index:</td>
<td>A C T G</td>
</tr>
<tr>
<td></td>
<td>C T G C</td>
</tr>
</tbody>
</table>
Step 3: Compressed Seeding

- Use the calculated shift amount to correctly extract seeds from the read sequence.
- Now both the reference genome and the read are half in length and their seeds can be still correctly matched.

Read sequence: 

Diet pattern: 

Patterned read: 

k-mers:

<table>
<thead>
<tr>
<th>G</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>A</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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

| C | T | G | C |
Step 4: Location Voting

- Seeds are *sparsified* and thus cannot be **directly chained**, instead to detect mapping locations we use **the number of matching seeds in a region**

**Genome sequence:**

**Read sequence:**
Step 5: Sequence Alignment

Dynamic programming matrix

nature computational science

Explore content  About the journal  Publish with us

nature  nature computational science  brief communications  article

Brief Communication | Published: 28 February 2022

Accelerating minimap2 for long-read sequencing applications on modern CPUs

Saurabh Kalikar, Chirag Jain, Md Vasimuddin & Sanchit Misra

Nature Computational Science 2, 78–83 (2022) | Cite this article

BMC Bioinformatics

Home  About  Articles  Submission Guidelines  Join The Board

Methodology  Open Access  Published: 19 February 2018

Introducing difference recurrence relations for faster semi-global alignment of long sequences

Hajime Suzuki & Masahiro Kasahara

BMC Bioinformatics 19, Article number: 45 (2018) | Cite this article

7719 Accesses  39 Citations  66 Altmetric | Metrics
CPU Implementation

 Genome-on-Diet is implemented on top of minimiap2 (2.24-r1122 version as of 11 November 2022)
Introducing Four Optimization Strategies

- **Accelerating Indexing & Seeding with SIMD Instructions**
  - Calculating $8 \times (\frac{512}{32\times2})$ overlapping k-mers along with their hash values in parallel

- **Sorting Seed Locations**
  - Merge sort instead of Radix and Heap sort algorithms

- **Rescuing Mapping Location**
  - Based on two voting thresholds

- **Handling Exactly-Matching Short Reads**
  - It is observed that 80% of short reads usually exactly match to the reference genome
Applications of Sparsified Genomics

- **Applications that compare sequences for similarity**
  - Genome similarity & genomic distance
  - Prealignment filtering
  - Containment search

- **Applications that generate huge index**
  - Taxonomic profiling
  - Pangenomics

- **Applications that require building index during the analysis**
  - Read mapping for many assembly versions of the same species
  - Identifying *de novo* variations by comparing family members
  - Identifying somatic variations by comparing healthy and tumor cells of the same patient
  - And many more ...

They may or may not require building index

Index can be up to 21.25x larger in size than a single (2-bit encoded) indexed genome

Indexing and seeding time account for

97% Taxonomic Profiling

10%-27% Read Mapping
Genome-on-Diet vs. Spaced Seeding?

Spaced Seeding:

ACCCCTAACCCTAACCCCTAACCCTAA...
ACCCCTAACCC
CCCTAACCCCT
CCTAACCCCTA
CTAACCCCTAA
TAACCTAAC
AACCTAAC

Genome-on-Diet:

ACCCCTAACCTAACCCCTAACCCCTAA...
100010100110001010011000101001
A__T_A__CT___C_T__CC___A...
A__T_A__C
T_A__CT
A__CT___C
A__CT___C

Increased execution time
No effect on peak memory footprint
No effect on the number of seeds
All seeds have the same pattern

Reduced execution time
Reduced peak memory footprint
Reduced number of seeds
Each seed may have its own pattern
Increasing Common k-mers Rate

| ALL | Minimizers | Spaced | Genome-on-Diet |

Genome-on-Diet provides the same or higher sensitivity compared to spaced seeding, while Genome-on-Diet is always faster and more memory efficient.
Evaluation Results
Great Benefits by Sparsified Genomics

Read Mapping
- **Genome-on-Diet** is 1.13-6.28x faster and has 2.1x smaller memory footprint, and 2x smaller index size compared to minimap2, for performing read mapping.

Containment Search
- **Genome-on-Diet** is 72.7-75.88x faster and 723.3x more storage-efficient than KMC3 combined with CMash, for performing containment search.

Metagenomic Profiling
- **Genome-on-Diet** is 54.15-61.88x faster and 720x more storage-efficient than Metalign, for performing taxonomic profiling of metagenomic samples.
Effect of Using Different Patterns

For performing both containment indexing and k-mer intersection

The performance scales linearly with the number of zeros determined in the pattern sequence
Genome-on-Diet performance is not affected by the value of minimizer window.

Location voting step is much faster than seed chaining.
Genome-on-Diet leads to the detection of a higher number of SNPs, indels, and SVs compared to minimap2
Preprint and Source Code

https://arxiv.org/abs/2211.08157

https://github.com/CMU-SAFARI/Genome-on-Diet

 Genome-on-Diet: Taming Large-Scale Genomic Analyses via Sparsified Genomics

Mohammed Alser, Julien Eudine, Onur Mutlu

https://arxiv.org/abs/2211.08157

https://github.com/CMU-SAFARI/Genome-on-Diet
Contributors

Mohammed Alser

Julien Eudine

Onur Mutlu

https://safari.ethz.ch
Genome-on-Diet
Taming Large-Scale Genomic Analyses via Sparsified Genomics

Mohammed Alser, PhD
@mealser

RECOMB 2023 – BIO-Arch
14 April 2023