Mohammed Alser

- **Senior Researcher and Lecturer**, SAFARI Research Group, ETH Zürich, since Sept. 2018.
- PhD from Bilkent University (Turkey) 2018, worked at UCLA, TU Dresden, and PETRONAS.
- **PhD these** in accelerating genome analysis, advisors: Can Alkan and Onur Mutlu, awarded:
  - IEEE Turkey Doctoral Dissertation Award
  - TÜBİTAK doctoral fellowship
  - The Best Palestinian PhD Student in Turkey
  - HiPEAC Collaboration Grant
- **ALSERM@ethz.ch**, [https://mealser.github.io/](https://mealser.github.io/), [https://twitter.com/mealser](https://twitter.com/mealser)

- My main research is in bioinformatics, computational genomics, metagenomics, and computer architecture.
- I am especially excited about **building** new data structures, algorithms, and architectures that **make intelligent genome analysis a reality**.
Example Paper Presentation III
Let’s Review This Paper [Alser+, Bioinformatics 2017]

Mohammed Alser, Hasan Hassan, Hongyi Xin, Oguz Ergin, Onur Mutlu, and Can Alkan
"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"
*Bioinformatics*, [published online, May 31], 2017.
[Source Code]
[Online link at Bioinformatics Journal]
GateKeeper: Accelerating Pre-Alignment in DNA Read Mapping

Mohammed Alser\textsuperscript{1}, Hasan Hassan\textsuperscript{2,3}, Hongyi Xin\textsuperscript{4}, Oğuz Ergin\textsuperscript{2}, Onur Mutlu\textsuperscript{1,3,4}, Can Alkan\textsuperscript{1}

Bioinformatics, 2017
Background, Problem, & Goal
Genomic analysis

Genomic analysis is the identification, measurement or comparison of genomic features such as DNA sequence, structural variation, gene expression, or regulatory and functional element annotation at a genomic scale. Methods for genomic analysis typically require high-throughput sequencing or microarray hybridization and bioinformatics.

https://onlinelearning.hms.harvard.edu/hmx/courses/genetic-testing/
https://www.nature.com/subjects/genomic-analysis
Genome Analysis

NO machine can read the entire content of a genome
Genome Sequencer is a Chopper

Regardless the sequencing machine, reads still lack information about their order and location (which part of genome they are originated from)
Reference Genome

.FASTA file:

>NG_008679.1:5001-38170 Homo sapiens paired box 6 (PAX6)
ACCCCTTTTTCTTATCATGATATTAAACTCTGGGCAGGTCCTGCGTAGAACCGGCTGTGATCT
GCCACTTTCCCTGCCAGCAGCGGCTGAGAATTGTGGGGAACCCGCCCTGCCAGGCCTCAGCCTG
CCTCCGCTCCCAGGTAACCCCGCCCGCTCCGCCGCCGCCGCTCGGCGGCGGCCGGGCTCGACGCTG
CCAGCGACTGCTGCTGCCCCAAATCAAGCCGCCCCAAATGGCCCCCCGGGCTTTAGTTTTGGCTTTAAAAG
GAGGCATACAAAGATGGAAAGCAGGTCATTGAGGGAGGGATGAGGAAGGGGGGTTGGAGGAGGACTTGTCCT
TGCCGAGTGCTGCTTCTTGCAAAAATGGCTTTCACTCCTAAAGATGGACTCCAGTGCTCCAGGGGCCT
GAGCTGGGAGTGGGCGCGGAGTCTGCTGCTGCTGCTGCTAAGCCACTCGCGACCAGCGAAAAATGCA
GGAGGTGGGAGACGCACTTGTGCTTCCAGGACCCCTCTGAGACAGCAGACATCCACGCTTTGGGAAAG
TCCGTACCAGGCCTGGAGCGCTTAAAGACACCTCTGCCGCGGCTGCGGCGGAGGTGCAGCAAGTGTTCCC
GCAGGGTCGAAGAGTGTCAGGTGGCTGGACCAGCAACAAAGTCTAGAGATGGGTCTTTGTTCTCAGAAAGACGC
GRCh38.p13

Description: Genome Reference Consortium Human Build 38 patch release 13 (GRCh38.p13)

Organism name: Homo sapiens (human)

Date: 2019/02/28

3,099,706,404 bases

Compressed .fna file (964.9 MB)

Genomic Reads

.FASTQ file:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>@HWI-EAS209_0006_FC706VJ:5:58:5894:21141#ATCAGC/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>TTAATGGATAATAAAATCTCCTAATAGCTTAGATNTACCTNNNNTAGTTTCTTGGAGA</td>
</tr>
<tr>
<td>+ sign &amp; identifier</td>
<td>+HWI-EAS209_0006_FC706VJ:5:58:5894:21141#ATCACG/1</td>
</tr>
<tr>
<td>Quality scores</td>
<td>efccccffccffceeefccffffffdf<code>feed]\</code><em>Ba</em>_[YBBB BBBB BBRTT]][]d____</td>
</tr>
</tbody>
</table>

Base T
phred Quality = 29
Obtaining .fastq Files


**ERX215261**: Whole Genome Sequencing of human TSI NA20754

1 ILLUMINA (Illumina HiSeq 2000) run: 4.1M spots, 818.7M bases, 387.2Mb downloads

**Design**: Illumina sequencing of library 6511095, constructed from sample accession SRS001721 for study accession SRP000540. This is part of an Illumina multiplexed sequencing run (9340_1). This submission includes reads tagged with the sequence TTAGGCAT.

**Submitted by**: The Wellcome Trust Sanger Institute (SC)

**Study**: Whole genome sequencing of (TSI) Toscani in Italia HapMap population

- [PRJNA33847](https://www.ncbi.nlm.nih.gov/bioproject/PRJNA33847) • [SRP000540](https://www.ncbi.nlm.nih.gov/sra/SRP000540) • All experiments • All runs

**Sample**: Coriell GM20754

- [SAMN00001273](https://www.ncbi.nlm.nih.gov/bio-sample/SAMN00001273) • SRS001721 • All experiments • All runs

**Organism**: Homo sapiens

**Library**:
- **Name**: 6511095
- **Instrument**: Illumina HiSeq 2000
- **Strategy**: WGS
- **Source**: GENOMIC
- **Selection**: RANDOM
- **Layout**: PAIRED
- **Construction protocol**: Standard

**Runs**: 1 run, 4.1M spots, 818.7M bases, 387.2Mb

<table>
<thead>
<tr>
<th>Run</th>
<th># of Spots</th>
<th># of Bases</th>
<th>Size</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR240727</td>
<td>4,093,747</td>
<td>818.7M</td>
<td>387.2Mb</td>
<td>2013-03-22</td>
</tr>
</tbody>
</table>

COVID-19 is an emerging, rapidly evolving situation.
Solving the Puzzle

Reference genome

Reads

Map reads to a known reference genome with some minor differences allowed.
Bottlenecked in Read Mapping!!

48 Human whole genomes
at 30 × coverage
in about 2 days

Human genome
32 CPU hours
on a 48-core processor

Illumina NovaSeq 6000

What makes read mapper slow?
Let’s first learn how to map a read
Matching Each Read with Reference Genome

.FASTA file:

>NG_008679.1:5001-38170 Homo sapiens paired box 6 (PAX6)
ACCCAGGATACCTATTATTTAATCTCTGGGCGGCGACCTTTGTCATAGCTGCTGCTGGGAACGCCGCTGCTGCAGGCTACAACCTGCTGCCCTCCGC
CTCCGGCTCCAGGTAAAAGCCGCGCCCGCCCGCTGCGGGGCGCGCGGGGCGGGGCGGCTCTCGGGCGCGCGGGCGGGGGTTGATTTTGCTTTTTAAAA
GAGGCGATAAAGATGGAAGCGAGTTACTGAGGAGGGGAAATGGAAGGAGGGCTGGAGGAGGAGGACCTTGCTTT
TCGGAGTGTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
Base-by-Base Comparison

read 1: CCT TAG TAT TAG TAC GT T

ref 1: CGT TAG CTA TAT AAT CG TAC GAT

reference segment that spans locations (5, 7, and 9)
Edit distance is defined as the minimum number of edits (i.e. insertions, deletions, or substitutions) needed to make the read exactly matches the reference segment.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>o - - r g a n i z a t i o n</td>
<td>o - - r g a n i z a t i o n</td>
</tr>
<tr>
<td>o p e r - - - - - - a - - - t i o n</td>
<td>o p e r - - - - - - a - - - t i o n</td>
</tr>
</tbody>
</table>

Edit distance = 7

<table>
<thead>
<tr>
<th>Ref</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>o - - r g a n i z a t i o n</td>
<td>o - - r g a n i z a t i o n</td>
</tr>
<tr>
<td>o p e r - a - - - - - - t i o n</td>
<td>o p e r - a - - - - - - t i o n</td>
</tr>
</tbody>
</table>

Edit distance = 4
What Makes Read Mapper Slow?

Key Observation # 1

93% of the read mapper’s execution time is spent in sequence alignment.

Alser et al, Bioinformatics (2017)
What Makes Read Mapper Slow? (cont’d)

Key Observation # 2

98% of candidate locations have high dissimilarity with a given read.

What Makes Read Mapper Slow? (cont’d)

Key Observation # 3

- **Quadratic-time** dynamic-programming algorithm **WHY?!**
  - Enumerating all possible prefixes

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>E</th>
<th>T</th>
<th>H</th>
<th>E</th>
<th>R</th>
<th>L</th>
<th>A</th>
<th>N</th>
<th>D</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</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>

- NETHERLANDS x SWITZERLAND
- NETHERLANDS x S
- NETHERLANDS x SW
- NETHERLANDS x SWI
- NETHERLANDS x SWIT
- NETHERLANDS x SWITZ
- NETHERLANDS x SWITZE
- NETHERLANDS x SWITZER
- NETHERLANDS x SWITZERL
- NETHERLANDS x SWITZERLA
- NETHERLANDS x SWITZERLAN
- NETHERLANDS x SWITZERLAND
What Makes Read Mapper Slow? (cont’d)

Key Observation # 3

- **Quadratic-time** dynamic-programming algorithm
  - Enumerating all possible prefixes

- **Data dependencies** limit the computation parallelism
  - Processing row (or column) after another

- **Entire matrix** is computed even though strings can be dissimilar.
  - Number of differences is computed only at the backtracking step.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>E</th>
<th>T</th>
<th>H</th>
<th>E</th>
<th>R</th>
<th>L</th>
<th>A</th>
<th>N</th>
<th>D</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>W</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>I</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>T</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Z</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>R</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>L</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>N</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
Read Mapping in 111 pages!

Analyzing 107 read mappers (1988-2020) in depth


Quantitative Biology > Genomics

[Submitted on 28 Feb 2020 (v1), last revised 9 Jul 2020 (this version, v3)]

Technology dictates algorithms: Recent developments in read alignment

Mohammed Alser, Jeremy Rotman, Kodi Taraszka, Huwenbo Shi, Pelin Icer Baykal, Harry Taegyun Yang, Victor Xue, Sergey Knyazev, Benjamin D. Singer, Brunilda Balliu, David Koslicki, Pavel Skums, Alex Zelikovsky, Can Alkan, Onur Mutlu, Serghei Mangul

Alser+, "Technology dictates algorithms: Recent developments in read alignment", arXiv, 2020

GitHub: https://github.com/Mangul-Lab-USC/review_technology_dictates_algorithms
Goal: Minimizing Alignment Time

Sequence Alignment is expensive

Our goal is to accelerate read mapping by reducing the need for dynamic programming algorithms
Novelty, Key Approach, and Ideas
Key Idea

- **Genomic Strings**
  - **Dissimilar Strings**
    - Ignore as number of differences exceeds a threshold.
  - **Similar Strings**
    - Find number and location of differences?

**EXPENSIVE!**
Key observation:
- If two strings differ by $E$ edits, then every pairwise match can be aligned in at most $2E$ shifts.

Key ideas:
- Compute “Shifted Hamming Distance”: AND of $2E+1$ Hamming vectors of two strings, to identify invalid mappings.
- Use only bit-parallel operations that nicely map to:
  - SIMD instructions
  - FPGA
  - Logic layer of the 3D-stacked memory
  - In-memory accelerators (e.g., Ambit)
Proposed Solution: GateKeeper

1st FPGA-based Alignment Filter

Pre-Alignment Filter

FPGA-based Alignment Filter

x$10^{12}$ mappings

Query the Index

x$10^3$ mappings

High throughput DNA sequencing (HTS) technologies

Read Pre-Alignment Filtering
Fast & Low False Positive Rate

Read Alignment
Slow & Zero False Positives
1. Filter out most of incorrect mappings.
2. Preserve all correct mappings.
3. Do it quickly.
Mechanisms (in some detail)
Mechanisms

- **Key observation:**
  - If two strings differ by $E$ edits, then every pairwise match can be aligned in at most $2E$ shifts.
Hamming Distance ($\sum \Theta$)

3 matches 5 mismatches

*Edit = 1 Deletion*

To cancel the effect of a deletion, we need to shift in the *right* direction
Shifted Hamming Distance (Xin+ 2015)

![Diagram of Shifted Hamming Distance](image)

- XOR
- AND
- Count 1's

**Examples:**
- 0 0 0 1 1 1 1 1
- 1 1 1 0 0 0 0 0

- 7 matches
- 1 mismatches

**Edit = 1 Deletion**
Mechanisms

- **Key observation:**
  - If two strings differ by $E$ edits, then every pairwise match can be aligned in at most $2E$ shifts.

- **Key ideas:**
  - Compute “Shifted Hamming Distance”: AND of $2E+1$ Hamming vectors of two strings, to identify invalid mappings
Our goal to track the diagonally consecutive matches in the neighborhood map.
Independent vectors can be processed in parallel using hardware technologies
Hardware Architecture
GateKeeper Walkthrough (cont’d)

- E right-shift registers (length=ReadLength)
- E left-shift registers (length=ReadLength)
- \((2E+1) \times \text{(ReadLength)}\) 2-XOR operations.

AND all masks, ACCEPT iff number of ‘1’ ≤ Threshold

- \((2E)\times\text{(ReadLength)}\) 2-AND operations.
- \((\text{ReadLength}/4)\) 5-input LUT.
- \(\log_2\text{ReadLength}\)-bit counter.

Hamming mask after amending

- \((2E+1)\times\text{(ReadLength)}\) 5-input LUT.
Configurable logic blocks (CLBs) are the main logic resources for implementing sequential as well as combinatorial circuits.

The LUTs in 7 series FPGAs can be configured as either a 6-input LUT with one output, or as two 5-input LUTs with separate outputs.

Figure 1-1: Arrangement of Slices within the CLB

Table 2-1: Logic Resources in One CLB

<table>
<thead>
<tr>
<th>Slices</th>
<th>LUTs</th>
<th>Flip-Flops</th>
<th>Arithmetic and Carry Chains</th>
<th>Distributed RAM</th>
<th>Shift Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>16</td>
<td>2</td>
<td>256 bits</td>
<td>128 bits</td>
</tr>
</tbody>
</table>

"7 Series FPGAs Configurable Logic Block", User Guide, Xilinx 2016
Key Results:
Methodology and Evaluation
Methodology

- **System setup:**
  - 3.6 GHz Intel i7-3820 (supports only PCIe 2.0)
  - Xilinx VC709 (~$5000)
    - Architecture implementation using Vivado 2014.4 in Verilog
    - RIFFA 2.2 to perform Host-FPGA PCIe communication

- **Evaluated dataset:**
  - **Real** sequencing read set (ERR240727_1.fastq)
  - Five **simulated** read sets of 100 bp and 300 bp long Illumina-like reads with different type and number of edits.
Prior Work on Pre-Alignment Filtering

- **Adjacency Filter** *(BMC Genomics, 2013)*
  - Slow
  - Accepts a large number of dissimilar sequences.

- **Shifted Hamming Distance** *(SHD) (Bioinformatics, 2015)*
  - It requires the same execution time as the Adjacency Filter
  - It accepts 4X fewer dissimilar sequences compared to the Adjacency Filter.
  - It suffers from a limited sequence length (≤ 128 bp)
VC709 Resource Utilization

Theoretically:

- Up to **140 cores** on a single FPGA (E=5, 100bp)
- BUT **bottlenecked** by PCIe bandwidth
- **Small area allows integration** into FPGAs already inside of sequencers

<table>
<thead>
<tr>
<th>Table 2. FPGA resource utilization for a single GateKeeper core</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read length</strong></td>
</tr>
<tr>
<td>Edit distance</td>
</tr>
<tr>
<td>Slice LUT(^a)</td>
</tr>
<tr>
<td>Slice Register(^b)</td>
</tr>
</tbody>
</table>

\(^a\)LUT: look-up tables.
\(^b\)Flip-flop.
Experimentally:

- *GateKeeper* aligns each read against up to 8 and 16 different reference segments in parallel, without violating the timing constraints for a sequence lengths of 300 and 100 bp, respectively.

### Table 3. Overall system resource utilization under different read lengths and edit distance thresholds

<table>
<thead>
<tr>
<th>Read length</th>
<th>100 bp</th>
<th>300 bp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 GateKeeper cores</td>
<td>8 GateKeeper cores</td>
</tr>
<tr>
<td>Edit distance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>Slice LUT</td>
<td>32%</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>45%</td>
<td>69%</td>
</tr>
<tr>
<td>Slice register</td>
<td>2%</td>
<td>17%</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>91%</td>
</tr>
<tr>
<td>Block memory</td>
<td>2%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>2%</td>
</tr>
</tbody>
</table>
**GateKeeper Accelerator Architecture**

- **Maximum data throughput** = \( \sim 13.3 \text{ billion bases/sec} \)
- Can examine 8 (300 bp) or 16 (100 bp) mappings *concurrently* at 250 MHz
- Occupies 50% (100 bp) to 91% (300 bp) of the FPGA slice LUTs and registers

---

![Diagram of GateKeeper Accelerator Architecture](image)
FPGA Chip Layout

GateKeeper: 17.6%, PCIe Controller, RIFFA, and IO: 5%

300 bp
E=15
**Speed & Accuracy Results**

90x-130x faster
than SHD (Xin et al., 2015) and the Adjacency Filter (Xin et al., 2013).

Accepts 4x fewer dissimilar strings
than the Adjacency Filter (Xin et al., 2013).

10x speedup
with the addition of GateKeeper to the mrFAST mapper (Alkan et al., 2009).

Freely available online

[github.com/BilkentCompGen/GateKeeper](https://github.com/BilkentCompGen/GateKeeper)
Summary
Executive Summary

- **Problem:** There is a significant performance gap between high-throughput DNA sequencers and read mapper.

- **Observations:** Sequence alignment is computationally expensive and unavoidable.

- **Goal:** provide the first hardware accelerator architecture (as a pre-alignment filter) for quickly rejecting dissimilar sequences.

- **Key Results:**
  - Provides a huge speedup of up to 130x compared to the previous state of the art software solution.
GateKeeper Conclusions

- FPGA-based pre-alignment filtering greatly speeds up read mapping
  - 10x speedup of a state-of-the-art mapper (mrFAST)

- FPGA-based pre-alignment can be integrated with the sequencer
  - It can help to hide the complexity and details of the FPGA
  - Enables real-time filtering while sequencing
More on SHD (SIMD Implementation)

- Download and test for yourself
- https://github.com/CMU-SAFAIR/Shifted-Hamming-Distance

Sequence analysis

Shifted Hamming distance: a fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping

Hongyi Xin\textsuperscript{1,*}, John Greth\textsuperscript{2}, John Emmons\textsuperscript{2}, Gennady Pekhimenko\textsuperscript{1}, Carl Kingsford\textsuperscript{3}, Can Alkan\textsuperscript{4,*} and Onur Mutlu\textsuperscript{2,*}
More on GateKeeper

- Download and test for yourself
  https://github.com/BilkentCompGen/GateKeeper

GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping

Mohammed Alser, Hasan Hassan, Hongyi Xin, Oğuz Ergin, Onur Mutlu, Can Alkan

Bioinformatics, Volume 33, Issue 21, 01 November 2017, Pages 3355–3363,
https://doi.org/10.1093/bioinformatics/btx342

Published: 31 May 2017   Article history

Alser+, "GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping", Bioinformatics, 2017.
Strengths
Strengths

- **New and simple** solution to a critical problem. New algorithm and hardware architecture.
- GateKeeper **does not sacrifice** any of the aligner capabilities, as it does not modify or replace the alignment step.
- Design is **scalable**; could add more processing cores in the future.
- Some sequencers use **FPGAs** as well, so GateKeeper could be integrated into them.
Strengths (cont’d)

- Authors understand and highlight limitations of GateKeeper
- Greatly improves filtering speed and accuracy
- Spurred quite a few papers that build on GateKeeper
- Well-written, interesting and easy to understand paper
Weaknesses
Recall: Try to Avoid Rat Holes

Performance Analysis Rat Holes

Workload  Metrics  Configuration  Details

Source: https://www.cse.wustl.edu/~jain/iuceed/ftp/k_10adp.pdf
Weaknesses

- The benefits of such a mechanism require an FPGA and advanced knowledge with computers, this may be problematic for some biologists/genomicists/geneticists.

- The amendment of the random zeros is a simple “hack” to reduce the number of false positives, but there is no explanation why GateKeeper only flips the patterns 101 and 1001, what about 10001? And 10^n1?

- The paper can be confusing at times due to the use of a “supplementary material” document that is constantly referred to (but understandable as there was a page limit set by the publication journal).
Weaknesses (cont’d)

- GateKeeper’s **accuracy degrades** exponentially for $E > 2\%$, and becomes ineffective for $E > 8\%$.

- GateKeeper is tested using short reads
  - 3rd generation sequencing machines produce much longer reads
Thoughts and Ideas
Extensions

- Can we improve the filtering accuracy
  - Don’t amend, count the number of matches accurately.
  - Yes, see MAGNET paper [Alser et al. arXiv preprint 2017]. But this requires large number of LUTs.
MAGNET [Alser+, arXiv 2017]


MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering

Alser, Mohammed; Mutlu, Onur; and Alkan, Can
MAGNET Walkthrough

Build Neighborhood Map
Track the Diagonally Consecutive Matches
ACCEPT iff number of ‘1’ ≤ Threshold

Find the longest segment of consecutive zeros
Exclude the errors from the search space
Divide the problem into two subproblems and repeat
Total number of edits = number of 1’s in MAGNET bit-vector
Extensions

- Can we improve the filtering accuracy
  - Don’t amend, count the number of matches accurately.
    - Yes, see MAGNET paper [Alser et al. arXiv preprint 2017]. But this requires large number of LUTs.

- Can we improve the filtering accuracy and scalability
  - Yes, see Shouji paper [Alser et al. Bioinformatics 2019].
[Source Code]
[Online link at Bioinformatics Journal]
## Shouji Walkthrough

**Building the Neighborhood Map**

- Finding all common subsequences (diagonal segments of consecutive zeros) shared between two given sequences.

**Storing it @ Shouji Bit-vector**

- ACCEPT iff number of ‘1’ ≤ Threshold

---

**Shouji: a fast and efficient pre-alignment filter for sequence alignment**, *Bioinformatics* 2019, https://doi.org/10.1093/bioinformatics/btz234
# Shouji Walkthrough

## Building the Neighborhood Map

Storing it @ Shouji Bit - vector

## Finding all common subsequences (diagonal segments of consecutive zeros) shared between two given sequences.


<table>
<thead>
<tr>
<th>i</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>1</td>
<td>G</td>
<td>G</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>A</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>G</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>G</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>G</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ACCEPT iff number of ‘1’ ≤ Threshold**
Extensions

- Can we improve the filtering accuracy
  - Don’t amend, count the number of matches accurately.
  - Yes, see MAGNET paper [Alser et al. *arXiv preprint 2017*]. But this requires large number of LUTs.

- Can we improve the filtering accuracy and scalability
  - Yes, see Shouji paper [Alser et al. *Bioinformatics 2019*].

- Can we solve the FPGA-CPU communication bottleneck?
  - *Where it makes sense*: Processing-in-memory, Processing-near-storage, Processing-while-sequencing?
  - Yes, see GRIM-Filter [Kim et al. *BMC Genomics 2018*].
GRIM-Filter [Kim+, BMC Genomics 2018]


Proceedings of the 16th Asia Pacific Bioinformatics Conference (APBC), Yokohama, Japan, January 2018.


---

GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies

Jeremie S. Kim$^{1,6,*}$, Damla Senol Cali$^1$, Hongyi Xin$^2$, Donghyuk Lee$^3$, Saugata Ghose$^1$, Mohammed Alser$^4$, Hasan Hassan$^6$, Oguz Ergin$^5$, Can Alkan$^{*4}$, and Onur Mutlu$^{*6,1}$
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: 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.
Integrating GRIM-Filter into a Read Mapper

**INPUT:** Read Sequence
GAACCTTGCAG ... GTATT

1. **GRIM-Filter:**
   Filter Bitmask Generator

   Seed Location Filter Bitmask
   ...0001010 ...011010 ...

2. **GRIM-Filter:**
   Seed Location Checker

   KEEP
   KEEP
   DISCARD

   INPUT: All Potential Seed Locations
   020128 020131 414415 ...

3. **Reference Segment Storage**

   Reference segment @ 020131 ...
   .
   .
   .
   Reference segment @ 414415 ...

4. **Read Mapper:**
   Sequence Alignment

   Edit-Distance Calculation

**OUTPUT:** Correct Mappings
Can We Do Better?

Faster, More Accurate, More Scalable

Pre-Alignment Filtering
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"

[Source Code]
[Online link at Bioinformatics Journal]
of value ‘0’) in its corresponding HRT. 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)

<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>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>2nd 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>1st 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>1st Lower Diagonal</td>
<td>0</td>
<td>1</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>
</tr>
<tr>
<td>2nd 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>3rd 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  Finding the Optimal Routing Path  Examining the Snake Survival

\[ E = 3 \]

<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></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>2\text{nd} 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>1\text{st} 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>1\text{st} 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>2\text{nd} 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>3\text{rd} 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>

\text{ENTRANCE}   \text{EXIT}
SneakySnake Walkthrough

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

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

This is what you actually need to **build** and it can be done **on-the-fly!**
FPGA Resource Analysis

- FPGA resource usage for a single filtering unit of GateKeeper, Shouji, and Snake-on-Chip for a sequence length of 100 and under different edit distance thresholds (E).

<table>
<thead>
<tr>
<th></th>
<th>E (bp)</th>
<th>Slice LUT</th>
<th>Slice Register</th>
<th>No. of Filtering Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GateKeeper</strong></td>
<td>2</td>
<td>0.39%</td>
<td>0.01%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.71%</td>
<td>0.01%</td>
<td>16</td>
</tr>
<tr>
<td><strong>Shouji</strong></td>
<td>2</td>
<td>0.69%</td>
<td>0.08%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.72%</td>
<td>0.16%</td>
<td>16</td>
</tr>
<tr>
<td><strong>Snake-on-Chip</strong></td>
<td>2</td>
<td>0.68%</td>
<td>0.16%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.42%</td>
<td>0.34%</td>
<td>16</td>
</tr>
</tbody>
</table>
Filtering Accuracy

Fig. 10: The execution time of SneakySnake, Parasail, and SneakySnake integrated with Parasail using long sequences, (a) 10Kbp and (b) 100Kbp, and 40 CPU threads. 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 Parasail. We present the end-to-end speedup values obtained by integrating SneakySnake with Parasail.
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.
Takeaways
Key Takeaways

- A novel method to accelerate Sequence Alignment in genome analysis.
- Simple and effective
- Hardware/software cooperative
- Good potential for work building on it to extend it
  - To make things more efficient and effective
  - Multiple works have already built on the paper (see MAGNET, Shouji, GRIM-Filter, SneakySnake)
- Easy to read and understand paper
Open Discussion
Discussion Starters (I)

- Thoughts on the previous ideas?

- Rethinking Alignment and Pre-alignment?
  - Re-use the results of the pre-alignment filter?
  - Improve the accuracy of pre-alignment filtering to achieve an optimal alignment?

- Extend the solution to longer reads, higher edit distance thresholds?

- Is this solution clearly advantageous in some cases?
Discussion Starters (II)

- Data movement is still a bottleneck. How could we try to reduce it?
  - Placing the accelerator closer to memory
  - Using newer and faster I/O
  - Closely integrate the accelerator into sequencers for real-time pre-alignment filtering
  - Offer cloud computing with access to advanced FPGA chips
Discussion Starters (III)

- Can you think of fields that could be similarly in need of string alignment as read mapping in bioinformatics?
- Natural language processing
  - OCR error correction
  - Autocorrection in text-based editors or apps
  - Reconstruction of languages using the comparative method
  - Social sciences

Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem

François Clautiaux\textsuperscript{a,b,*}, Ruslan Sadykov\textsuperscript{b,a}, François Vanderbeck\textsuperscript{a,b}, Quentin Viaud\textsuperscript{a,b}

\textsuperscript{a}IMB, Université de Bordeaux, 351 cours de la Libération, 33405 Talence, France
\textsuperscript{b}INRIA Bordeaux - Sud-Ouest, 200 avenue de la Vieille Tour, 33405 Talence, France

Clautiaux+, "\textit{Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem}, Discrete Optimization, 2018."
More Details on GateKeeper [Alser+, Bioinformatics 2017]

Mohammed Alser, Hasan Hassan, Hongyi Xin, Oguz Ergin, Onur Mutlu, and Can Alkan
"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"
*Bioinformatics*, [published online, May 31], 2017.
[Source Code]
[Online link at Bioinformatics Journal]
GateKeeper: Accelerating Pre-Alignment in DNA Read Mapping

Mohammed Alser1, Hasan Hassan2,3, Hongyi Xin4, Oğuz Ergin2, Onur Mutlu1,3,4, Can Alkan1

Bioinformatics, 2017
What *else* can be *done*?
Accelerating Genome Analysis: Overview

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,

"Accelerating Genome Analysis: A Primer on an Ongoing Journey"


[Slides (pptx)(pdf)]
[Talk Video (1 hour 2 minutes)]
GenASM Framework [MICRO 2020]

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,

"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]
Problem & Our Goal

- Multiple steps of read mapping require approximate string matching
  - ASM enables read mapping to account for sequencing errors and genetic variations in the reads
- ASM makes up a significant portion of read mapping (more than 70%)
- One of the major bottlenecks of genome sequence analysis

**Our Goal:**
Accelerate approximate string matching by designing a fast and flexible framework, which can be used to accelerate multiple steps of the genome sequence analysis pipeline
GenASM: ASM Framework for GSA

Our Goal:
Accelerate approximate string matching
by designing a fast and flexible framework,
which can accelerate multiple steps of genome sequence analysis

- **GenASM:** *First* ASM acceleration framework for GSA
  - Based on the *Bitap* algorithm
    - Uses fast and simple bitwise operations to perform ASM
  - Modified and extended ASM algorithm
    - Highly-parallel Bitap with long read support
    - Bitvector-based novel algorithm to perform traceback
  - Co-design of our modified scalable and memory-efficient algorithms with low-power and area-efficient hardware accelerators
GenASM: Hardware Design

GenASM-DC: generates bitvectors and performs edit Distance Calculation

GenASM-TB: performs TraceBack and assembles the optimal alignment
GenASM: Hardware Design

Our specialized compute units and on-chip SRAMs help us to:

→ Match the rate of computation with memory capacity and bandwidth
→ Achieve high performance and power efficiency
→ Scale linearly in performance with the number of parallel compute units that we add to the system
GenASM-DC: Hardware Design

- **Linear cyclic systolic array** based accelerator
  - Designed to maximize parallelism and minimize memory bandwidth and memory footprint
Based on our synthesis of GenASM-DC and GenASM-TB accelerator datapaths using the Synopsys Design Compiler with a 28nm LP process:

- Both GenASM-DC and GenASM-TB operate @ 1GHz

### Key Results – Area and Power

<table>
<thead>
<tr>
<th></th>
<th>Area (mm²)</th>
<th>Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenASM-DC (64 PEs)</td>
<td>0.049</td>
<td>0.033</td>
</tr>
<tr>
<td>GenASM-TB</td>
<td>0.016</td>
<td>0.004</td>
</tr>
<tr>
<td>DC-SRAM (8 KB)</td>
<td>0.013</td>
<td>0.009</td>
</tr>
<tr>
<td>TB-SRAMs (64 x 1.5 KB)</td>
<td>0.055</td>
<td>0.055</td>
</tr>
</tbody>
</table>

**Total (1 vault):**
- Area: 0.334 mm²
- Power: 0.101 W

**Total (32 vaults):**
- Area: 10.69 mm²
- Power: 3.23 W

**% of a Xeon CPU core:**
- Area: 1%
- Power: 1%
Key Results – Area and Power

- Based on our synthesis of GenASM-DC and GenASM-TB accelerator datapaths using the Synopsys Design Compiler with a 28nm LP process:
  - Both GenASM-DC and GenASM-TB operate @ 1GHz

GenASM has low area and power overheads
Use Cases of GenASM

Reference genome

Indexing

Hash table based index

Seeding

Candidate mapping locations

Pre-Alignment Filtering

Remaining candidate mapping locations

Read Alignment

Optimal alignment

Reads from sequenced genome
Use Cases of GenASM (cont’d.)

(1) Read Alignment Step of Read Mapping
   - Find the optimal alignment of how reads map to candidate reference regions

(2) Pre-Alignment Filtering for Short Reads
   - Quickly identify and filter out the unlikely candidate reference regions for each read

(3) Edit Distance Calculation
   - Measure the similarity or distance between two sequences

- We also discuss other possible use cases of GenASM in our paper:
  - Read-to-read overlap finding, hash-table based indexing, whole genome alignment, generic text search
Key Results

(1) Read Alignment
- 116 × speedup, 37 × less power than Minimap2 (state-of-the-art SW)
- 111 × speedup, 33 × less power than BWA-MEM (state-of-the-art SW)
- 3.9 × better throughput, 2.7 × less power than Darwin (state-of-the-art HW)
- 1.9 × better throughput, 82% less logic power than GenAx (state-of-the-art HW)

(2) Pre-Alignment Filtering
- 3.7 × speedup, 1.7 × less power than Shouji (state-of-the-art HW)

(3) Edit Distance Calculation
- 22–12501 × speedup, 548–582 × less power than Edlib (state-of-the-art SW)
- 9.3–400 × speedup, 67 × less power than ASAP (state-of-the-art HW)
More on GenASM Framework [MICRO 2020]


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]
What if we got a new version of the reference genome?

Reference genome

Reads

AirLift [Kim+, arXiv 2021]

Jeremie S. Kim, Can Firtina, Meryem Banu Cavlak, Damla Senol Cali, Mohammed Alser, Nastaran Hajinazar, Can Alkan, Onur Mutlu
[Source Code]
[Online link at arXiv]
Key observation: Reference genomes are updated frequently. Repeating read mapping is a computationally expensive workload.

Key idea: Update the mapping results of only affected reads depending on how a region in the old reference relates to another region in the new reference.

Key results:
- reduces number of reads that needs to be re-mapped to new reference by up to 99.99%
- reduces overall runtime to re-map reads by 6.7x, 6.6x, and 2.8x for large (human), medium (C. elegans), and small (yeast) reference genomes
Clustering the Reference Genome Regions

**Fig. 2.** Reference Genome Regions.
Technology dictates algorithms: Recent developments in read alignment

Mohammed Alser, Jeremy Rotman, Kodi Taraszka, Huwenbo Shi, Pelin Icer Baykal, Harry Taegyun Yang, Victor Xue, Sergey Knyazev, Benjamin D. Singer, Brunilda Balliu, David Koslicki, Pavel Skums, Alex Zelikovsky, Can Alkan, Onur Mutlu, Serghei Mangul

Alser+, "Technology dictates algorithms: Recent developments in read alignment", arXiv, 2020

GitHub: https://github.com/Mangul-Lab-USC/review_technology_dictates_algorithms
Processing Genomic Data Where it Makes Sense

Intelligent Genome Analysis

FPGAs

Hybrid Main Memory

Heterogeneous Processors and Accelerators

(General Purpose) GPUs

Persistent Memory/Storage

Sequencing Machine
What is Intelligent Genome Analysis?

- Fast genome analysis
  - Real-time analysis

- Using intelligent architectures
  - Specialized HW with less data movement

- DNA is a valuable asset
  - Controlled-access analysis

- Population-scale genome analysis
  - Sequence anywhere at large scale!

- Avoiding erroneous analysis
  - E.g., your father is not your father

Bandwidth

Energy-efficiency & Latency

Privacy

Scalability

Accuracy
Achieving Intelligent Genome Analysis?

How and where to enable fast, accurate, cheap, privacy-preserving, and exabyte scale analysis of genomic data?
Most speedup comes from parallelism enabled by novel architectures and algorithms.
More on Fast Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Invited Lecture at Technion, Virtual, 26 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
More on Intelligent Genome Analysis …

Our Solution: GateKeeper

Alignment Filter + FPGA-based Alignment Filter = $10^{12}$ mappings

Low Speed & High Accuracy
Medium Speed, Medium Accuracy
High Speed, Low Accuracy

Safari

Computer Architecture - Lecture 8: Intelligent Genome Analysis (ETH Zürich, Fall 2020)

https://www.youtube.com/watch?v=ygmQpdDTL7o
Detailed Lectures on Genome Analysis

- Computer Architecture, Fall 2020, Lecture 3a
  - Introduction to Genome Sequence Analysis (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5

- Computer Architecture, Fall 2020, Lecture 8
  - Intelligent Genome Analysis (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14

- Computer Architecture, Fall 2020, Lecture 9a
  - GenASM: Approx. String Matching Accelerator (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=XoLpzmNPas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15

- Accelerating Genomics Project Course, Fall 2020, Lecture 1
  - Accelerating Genomics (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId

https://www.youtube.com/onurmutlulectures
Prior Research on Genome Analysis (1/2)

- Alser + "SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs.", *Bioinformatics*, 2020.


Alser+, "GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping", Bioinformatics, 2017.

Alser+, "MAGNET: understanding and improving the accuracy of genome pre-alignment filtering", IPSI Transaction, 2017.
Openings @ SAFARI

- We are hiring enthusiastic and motivated students and researchers at all levels.

- Join us now: safari.ethz.ch/apply
Thank you. Questions?
Seminar in Computer Architecture
Meeting 4: GateKeeper

Dr. Mohammed Alser
ALSERM@ethz.ch

ETH Zürich
Spring 2021
18 March 2021