SPRING: a next-generation compressor for FASTQ data

Shubham Chandak
Stanford University
Stanford Compression Workshop 2019
Joint work with

• Kedar Tatwawadi, Stanford University
• Idoia Ochoa, UIUC
• Mikel Hernaez, UIUC
• Tsachy Weissman, Stanford University
Outline

• Intro to genome sequencing
• FASTQ format and compression results
• SPRING algorithm
• SPRING as a practical tool
Genome sequencing

• Genome: long string of bases \{A, C, G, T\}
• Sequenced as noisy paired substrings (*reads*):

![Diagram of genome sequencing]

- Genome ~ 3 billion bases
- Coverage/Depth: ~30x-60x
Why compression?
Why compression?

500K human genomes

~1.5M eukaryote species
FASTQ format
FASTQ format

File 1
@ERR174324.1 HSQ1009_86:1:1101:1192:2116/1
ATTNGTCACTTCTCACCAGGCCCTCATTCACAACACTGGGAATTAATTCGAC...
+
CCCF#2ADHHHHJJJIIJJJIJJJJJJJGIJJJJJJJJJIIJJJJJJJJIIJJJJJJJ

File 2
@ERR174324.2 HSQ1009_86:1:1101:1192:2116/2
CAGANAGAGACTCTGTCTCAAAAAAAACAAAACAAACAAACAAAACAAAACAA
+
CCCF#2ADHFHHHJJIIJJJJJIIJJJJJJJJJJIIJJJJJJJJJJIIJJJJJJJJJJIIJJ

We’ll mostly focus on reads in this talk.
Read compression
Read compression

• For a typical 25x human dataset:
  • Uncompressed: 79 GB (1 byte/base)
Read compression

• For a typical 25x human dataset:
  • Uncompressed:  79 GB (1 byte/base)
  • Gzip:          ~20 GB (2 bits/base) – still far from optimal
Read compression

- For a typical 25x human dataset:
  - Uncompressed: 79 GB (1 byte/base)
  - Gzip: ~20 GB (2 bits/base) – still far from optimal
- Order of read pairs in FASTQ irrelevant – can this help?
### Read compression results

<table>
<thead>
<tr>
<th>Compressor</th>
<th>25x human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncompressed</td>
<td>79 GB</td>
</tr>
<tr>
<td>Gzip</td>
<td>~20 GB</td>
</tr>
</tbody>
</table>
## Read compression results

<table>
<thead>
<tr>
<th>Compressor</th>
<th>25x human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncompressed</td>
<td>79 GB</td>
</tr>
<tr>
<td>Gzip</td>
<td>~20 GB</td>
</tr>
<tr>
<td>FaStore (allow reordering)</td>
<td>6 GB</td>
</tr>
</tbody>
</table>
## Read compression results

<table>
<thead>
<tr>
<th>Compressor</th>
<th>25x human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncompressed</td>
<td>79 GB</td>
</tr>
<tr>
<td>Gzip</td>
<td>~20 GB</td>
</tr>
<tr>
<td>FaStore (allow reordering)</td>
<td>6 GB</td>
</tr>
<tr>
<td><strong>SPRING</strong> (no reordering)</td>
<td>3 GB</td>
</tr>
<tr>
<td><strong>SPRING</strong> (allow reordering)</td>
<td>2 GB</td>
</tr>
</tbody>
</table>
## Read compression results

<table>
<thead>
<tr>
<th>Compressor</th>
<th>25x human</th>
<th>100x human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncompressed</td>
<td>79 GB</td>
<td>319 GB</td>
</tr>
<tr>
<td>Gzip</td>
<td>~20 GB</td>
<td>~80 GB</td>
</tr>
<tr>
<td>FaStore (allow reordering)</td>
<td>6 GB</td>
<td>13.7 GB</td>
</tr>
<tr>
<td><strong>SPRING</strong> (no reordering)</td>
<td>3 GB</td>
<td>10 GB</td>
</tr>
<tr>
<td><strong>SPRING</strong> (allow reordering)</td>
<td>2 GB</td>
<td>5.7 GB</td>
</tr>
</tbody>
</table>
Key idea

- Storing reads equivalent to
Key idea

- Storing reads equivalent to
  - Store genome
Key idea

• Storing reads equivalent to
  • Store genome
  • Store read positions in genome
Key idea

- Storing reads equivalent to
  - Store genome
  - Store read positions in genome
  - Store noise in reads
Key idea

- Storing reads equivalent to
  - Store genome
  - Store read positions in genome
  - Store noise in reads

- Entropy calculations show this outperforms previous compressors
Key idea

• But... How to get the genome from the reads?
Key idea

- But... How to get the genome from the reads?
- Genome assembly too expensive - big challenges:
  - resolve repeats
  - get very long pieces of genome from shorter assemblies
Key idea

• But... How to get the genome from the reads?
• Genome assembly too expensive - big challenges:
  • resolve repeats
  • get very long pieces of genome from shorter assemblies
• Solution: Don’t need perfect assembly for compression!
SPRING workflow

Raw reads
SPRING workflow

Approximate assembly

Raw reads
SPRING workflow

- Raw reads
- Approximate assembly
- Assembled sequence
- Read position in assembled sequence
- Noisy bases
- Etc.
SPRING workflow

- Raw reads
- Approximate assembly
- Assembled sequence
- Read position in assembled sequence
- Noisy bases
- Etc.
- Encode
- BSC
- Compressed file
SPRING workflow

- Raw reads
- Approximate assembly
- Assembled sequence
- Read position in assembled sequence
- Noisy bases
- Etc.

In "allow reordering" mode: sort by position in approximate assembly

Compressed file
SPRING as a practical tool
SPRING as a practical tool

195 GB
25x human
FASTQ

2 hours
32 GB RAM
8 threads

7 GB
SPRING
archive

26 minutes
6 GB RAM
8 threads

Original
FASTQ
SPRING as a practical tool

- Support for:
  - Lossless and lossy modes
  - Variable length reads, long reads, etc.
  - Random access

195 GB
25x human
FASTQ

2 hours
32 GB RAM
8 threads

7 GB
SPRING
archive

26 minutes
6 GB RAM
8 threads

Original
FASTQ

2 hours
32 GB RAM
8 threads

7 GB
SPRING
archive

26 minutes
6 GB RAM
8 threads

Original
FASTQ
SPRING as a practical tool

- Support for:
  - Lossless and lossy modes
  - Variable length reads, long reads, etc.
  - Random access

- Github: [https://github.com/shubhamchandak94/SPRING/](https://github.com/shubhamchandak94/SPRING/)
SPRING as a practical tool

- Support for:
  - Lossless and lossy modes
  - Variable length reads, long reads, etc.
  - Random access

- Github: [https://github.com/shubhamchandak94/SPRING/](https://github.com/shubhamchandak94/SPRING/)

- Currently integrating with genie, an upcoming open source MPEG-G codec
Thank you!
References


• Shubham Chandak, Kedar Tatwawadi, Idoia Ochoa, Mikel Hernaez, Tsachy Weissman; SPRING: a next-generation compressor for FASTQ data, *Bioinformatics*, bty1015


• BSC: https://github.com/IlyaGrebnov/libbsc

• genie (open source MPEG-G codec): https://mitogen.github.io/

• Image credits:
  • https://www.genome.gov/27541954/dna-sequencing-costs-data/
  • https://twitter.com/nature/status/1050115893957730305
  • http://www.earlham.ac.uk/newsroom/decoding-life-earth