

Bowtie vs. SOAP/SOAP2

Efficient Alignment of Short Reads to the Human Genome

Jan Christian Kässens

Institut für Informatik
Christian-Albrechts-Universität zu Kiel

28 June 2010

Overview

Burrows-Wheeler Transformation

- Algorithm

- FM Index

- Summary

Bowtie

- INEXACTMATCH

- Phased Search

- Performance

SOAP

- SOAP

- SOAP2

- Parallel Performance

Comparison

References

Burrows-Wheeler Transformation

Michael Burrows & David Wheeler, 1994

- ▶ Michael Burrows & David Wheeler, 1994 at DEC Systems [1]
- ▶ data compression technique
 - ▶ finds sequence repetitions
- ▶ permutes a data block and creates index
- ▶ output is *slightly* larger
- ▶ fast searching while keeping the memory footprint small
 - ▶ used in bioinformatics since 2000

Burrows-Wheeler Transformation

Michael Burrows & David Wheeler, 1994

- ▶ Michael Burrows & David Wheeler, 1994 at DEC Systems [1]
- ▶ data compression technique
 - ▶ finds sequence repetitions
- ▶ permutes a data block and creates index
- ▶ output is *slightly* larger
- ▶ fast searching while keeping the memory footprint small
 - ▶ used in bioinformatics since 2000

Burrows-Wheeler Transformation

Michael Burrows & David Wheeler, 1994

- ▶ Michael Burrows & David Wheeler, 1994 at DEC Systems [1]
- ▶ data compression technique
 - ▶ finds sequence repetitions
- ▶ permutes a data block and creates index
- ▶ output is *slightly* larger
- ▶ fast searching while keeping the memory footprint small
 - ▶ used in bioinformatics since 2000

Burrows-Wheeler Transformation

Michael Burrows & David Wheeler, 1994

- ▶ Michael Burrows & David Wheeler, 1994 at DEC Systems [1]
- ▶ data compression technique
 - ▶ finds sequence repetitions
- ▶ permutes a data block and creates index
- ▶ output is *slightly* larger
- ▶ fast searching while keeping the memory footprint small
 - ▶ used in bioinformatics since 2000

Burrows-Wheeler Transformation

Michael Burrows & David Wheeler, 1994

- ▶ Michael Burrows & David Wheeler, 1994 at DEC Systems [1]
- ▶ data compression technique
 - ▶ finds sequence repetitions
- ▶ permutes a data block and creates index
- ▶ output is *slightly* larger
- ▶ fast searching while keeping the memory footprint small
 - ▶ used in bioinformatics since 2000

Burrows-Wheeler Transformation

An example

SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.BOXES

transforms to

TEXYDST.E.IXIXIXXSSMPPS.B..E.S.EUSFXDIIIOIIIT

Burrows-Wheeler Transformation

Forward Transformation

1. add marker \$ and calculate all cyclic rotations
2. sort them lexicographically
3. read result from last column

Example “assam”

assam

Burrows-Wheeler Transformation

Forward Transformation

1. add marker \$ and calculate all cyclic rotations
2. sort them lexicographically
3. read result from last column

Example “assam”

assam\$ →

 assam\$

 \$assam

 m\$assa

 am\$ass

 sam\$as

 ssam\$a

Burrows-Wheeler Transformation

Forward Transformation

1. add marker \$ and calculate all cyclic rotations
2. sort them lexicographically
3. read result from last column

Example “assam”

| | | | | |
|---------|---------|---------|---------|---------|
| | assam\$ | | \$assam | |
| | \$assam | | am\$ass | |
| assam\$ | → | m\$assa | → | assam\$ |
| | | am\$ass | | m\$assa |
| | | sam\$as | | sam\$as |
| | | ssam\$a | | ssam\$a |

Burrows-Wheeler Transformation

Forward Transformation

1. add marker \$ and calculate all cyclic rotations
2. sort them lexicographically
3. read result from last column

Example “assam”

| | | | | |
|---------|---------|---------|---------|-----------|
| | assam\$ | | \$assam | |
| | \$assam | | am\$ass | |
| assam\$ | → | m\$assa | → | assam\$ |
| | | am\$ass | | m\$assa |
| | | sam\$as | | sam\$a |
| | | ssam\$a | | ssam\$a |
| | | | | → ms\$asa |

Burrows-Wheeler Transformation

Reverse Transformation

Example

$L = \text{ms}\$ \text{asa}$

$F = \$ \text{aamss}$

$T =$

Be L the BW transformation of $T_{\text{original}}\$, L_i$ the $(i - 1)^{\text{th}}$ character in L .

1. $F := \text{sort}(L), j := 0, k := 0$
2. $T := \$$
3. do $|L| - 1$ times:
 - 3.1 $j := \text{CountOccurrences}(L_{0..k}, L_k)$
 - 3.2 $k := \text{FindJ}^{\text{th}}\text{Occurrence}(F, L_k, j)$
 - 3.3 $T := L_k T$

FM Index

suffix search on Burrows-Wheeler indices

- ▶ by Ferragina & Manzini, Washington D.C., 2001 [2]

Example on “acaacg\$” index

Pattern: aac

\$acaacg
aacg\$ac
acaacg\$
acg\$aca
caccg\$a
cg\$acaa
g\$acaac

FM Index

suffix search on Burrows-Wheeler indices

- ▶ by Ferragina & Manzini, Washington D.C., 2001 [2]

Example on “acaacg\$” index

Pattern: aac

```
$acaacg  
aacg$ac  
acaacg$  
acg$aca  
→ caccg$a  
→ cg$acaa  
g$acaac
```

FM Index

suffix search on Burrows-Wheeler indices

- ▶ by Ferragina & Manzini, Washington D.C., 2001 [2]

Example on “acaacg\$” index

Pattern: **aac**

```

$aacaacg
aacg$ac
aacaacg$
acg$aca
caacg$a
cg$aac
g$aacaac
  
```

FM Index

suffix search on Burrows-Wheeler indices

- ▶ by Ferragina & Manzini, Washington D.C., 2001 [2]

Example on “acaacg\$” index

Pattern: aac ✓

\$acaacg
 aacg\$a
 acaacg\$
 acg\$a
 caccg\$a
 cg\$a
 g\$a

FM Index

locating matches

Example on “acaacg\$” index

Pattern: aac

| | |
|----|----|
| \$ | g |
| a | c |
| a | \$ |
| a | a |
| c | a |
| c | a |
| g | c |

- ▶ Mark every m th row with corresponding position
- ▶ Look-up:
 - ▶ if a is at marked row → done
 - ▶ otherwise: apply LF-Mapping until marked row is found

FM Index

locating matches

Example on “acaacg\$” index

Pattern: aac

| | |
|----|----|
| \$ | g |
| a | c |
| a | \$ |
| a | a |
| c | a |
| c | a |
| g | c |

- ▶ Mark every m th row with corresponding position
- ▶ Look-up:
 - ▶ if a is at marked row → done
 - ▶ otherwise: apply LF-Mapping until marked row is found

Burrows-Wheeler Transformation & FM Indexing

Advantages

- ▶ small memory footprint
- ▶ fast lookup

Disadvantages

- ▶ slow index building
- ▶ complex data structures
- ▶ difficult for inexact queries

Burrows-Wheeler Transformation & FM Indexing

Advantages

- ▶ small memory footprint
- ▶ fast lookup

Disadvantages

- ▶ slow index building
- ▶ complex data structures
- ▶ difficult for inexact queries

Bowtie

Overview

- ▶ Langmead, et. al., University of Maryland, 2009 [4]

Bowtie algorithm

1. BW-transform reference sequence
 2. do alignment with INEXACTMATCH
 3. resolve original positions of matches
- ▶ pre-built indices freely available

Bowtie

Overview

- ▶ Langmead, et. al., University of Maryland, 2009 [4]

Bowtie algorithm

1. BW-transform reference sequence
 2. do alignment with INEXACTMATCH
 3. resolve original positions of matches
- ▶ pre-built indices freely available

Bowtie

Overview

- ▶ Langmead, et. al., University of Maryland, 2009 [4]

Bowtie algorithm

1. BW-transform reference sequence
 2. do alignment with INEXACTMATCH
 3. resolve original positions of matches
- ▶ pre-built indices freely available

Bowtie

Overview

- ▶ Langmead, et. al., University of Maryland, 2009 [4]

Bowtie algorithm

1. BW-transform reference sequence
 2. do alignment with INEXACTMATCH
 3. resolve original positions of matches
- ▶ pre-built indices freely available

Bowtie

Overview

- ▶ Langmead, et. al., University of Maryland, 2009 [4]

Bowtie algorithm

1. BW-transform reference sequence
 2. do alignment with INEXACTMATCH
 3. resolve original positions of matches
- ▶ pre-built indices freely available

Bowtie

INEXACTMATCH

Direct FM index search does not allow *mismatches*.

- ▶ establish an alignment policy
 - ▶ have a *quality value* assigned to each base, “Phred Score”
 - ▶ allow mismatches at low-quality bases
 - ▶ enforce minimum overall quality threshold *or fail*

Algorithm

1. perform regular FM index search.
2. select already-matched position
3. substitute base introducing a mismatch

Bowtie

INEXACTMATCH

Direct FM index search does not allow *mismatches*.

- ▶ establish an alignment policy
 - ▶ have a *quality value* assigned to each base, “Phred Score”
 - ▶ allow mismatches at low-quality bases
 - ▶ enforce minimum overall quality threshold *or fail*

Algorithm

1. perform regular FM index search.
2. select already-matched position
3. substitute base introducing a mismatch

Bowtie

INEXACTMATCH

Direct FM index search does not allow *mismatches*.

- ▶ establish an alignment policy
 - ▶ have a *quality value* assigned to each base, “Phred Score”
 - ▶ allow mismatches at low-quality bases
 - ▶ enforce minimum overall quality threshold *or fail*

Algorithm

1. perform regular FM index search.
2. select already-matched position
3. substitute base introducing a mismatch

Bowtie

INEXACTMATCH

Direct FM index search does not allow *mismatches*.

- ▶ establish an alignment policy
 - ▶ have a *quality value* assigned to each base, “Phred Score”
 - ▶ allow mismatches at low-quality bases
 - ▶ enforce minimum overall quality threshold *or fail*

Algorithm

1. perform regular FM index search.
2. select already-matched position
3. substitute base introducing a mismatch

Bowtie

INEXACTMATCH

Direct FM index search does not allow *mismatches*.

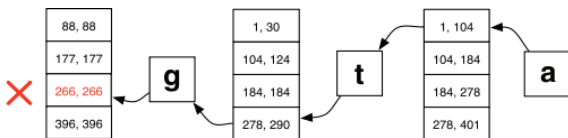
- ▶ establish an alignment policy
 - ▶ have a *quality value* assigned to each base, “Phred Score”
 - ▶ allow mismatches at low-quality bases
 - ▶ enforce minimum overall quality threshold *or fail*

Algorithm

1. perform regular FM index search.
2. select already-matched position
3. substitute base introducing a mismatch

Bowtie

INEXACTMATCH



“ggta” example

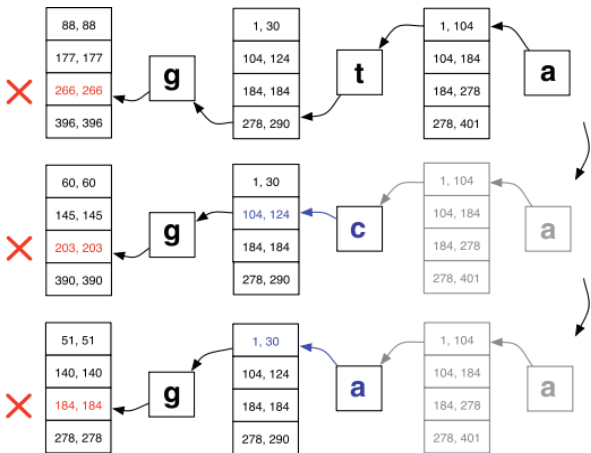
1. EXACTMATCH fails at position 3
2. substitute least-quality base
3. backtrack

Bowtie

INEXACTMATCH

"ggta" example

1. EXACTMATCH fails at position 3
2. substitute least-quality base
3. backtrack

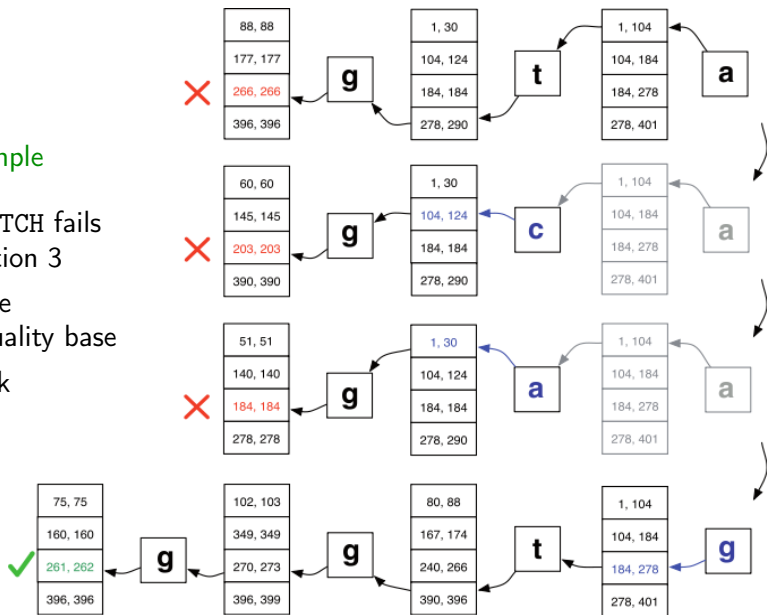


Bowtie

INEXACTMATCH

"ggta" example

1. EXACTMATCH fails at position 3
2. substitute least-quality base
3. backtrack



Bowtie

Excessive Backtracking

Problem

- ▶ Backtracking could lead to excessive stack and runtime growth on long reads

Solutions

- ▶ restrict backtracking steps (default: 125), then fail
- ▶ create *mirror index* and backtrack two times
 - ▶ first: allow substitutions only on 5' (high-quality) half
 - ▶ second: allow substitutions only on 3' (low-quality) half

Bowtie

Excessive Backtracking

Problem

- ▶ Backtracking could lead to excessive stack and runtime growth on long reads

Solutions

- ▶ restrict backtracking steps (default: 125), then fail
- ▶ create *mirror index* and backtrack two times
 - ▶ first: allow substitutions only on 5' (high-quality) half
 - ▶ second: allow substitutions only on 3' (low-quality) half

Bowtie

Excessive Backtracking

Problem

- ▶ Backtracking could lead to excessive stack and runtime growth on long reads

Solutions

- ▶ restrict backtracking steps (default: 125), then fail
- ▶ create *mirror index* and backtrack two times
 - ▶ first: allow substitutions only on 5' (high-quality) half
 - ▶ second: allow substitutions only on 3' (low-quality) half

Bowtie

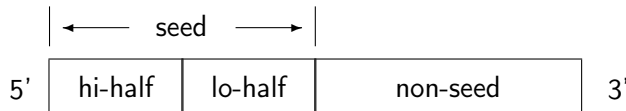
Phased Maq-like Search

Terminology

seed: high-quality half of the read (usually the 5' end)

hi-half: high-quality half of the seed (towards 5')

lo-half: low-quality half of the seed



Bowtie

Phased Maq-like Search (cont'd)

Cases with two mismatches allowed:

| | | | |
|----|-----|-----|--|
| 1. | 0 | 0 | |
| 2. | 0 | 1-2 | |
| 3. | 1-2 | 0 | |
| 4. | 1 | 1 | |

Search phases

1. find alignments for cases 1 and 2 (mirror)
2. find *partial* alignments for case 3 (forward)
3. use partial alignments and make full alignments for case 3 and 4 (mirror)

Bowtie

Phased Maq-like Search (cont'd)

Cases with two mismatches allowed:

| | | | |
|----|-----|-----|--|
| 1. | 0 | 0 | |
| 2. | 0 | 1-2 | |
| 3. | 1-2 | 0 | |
| 4. | 1 | 1 | |

Search phases

1. find alignments for cases 1 and 2 (mirror)
2. find *partial* alignments for case 3 (forward)
3. use partial alignments and make full alignments for case 3 and 4 (mirror)

Bowtie

Phased Maq-like Search (cont'd)

Cases with two mismatches allowed:

| | | | |
|----|-----|-----|--|
| 1. | 0 | 0 | |
| 2. | 0 | 1-2 | |
| 3. | 1-2 | 0 | |
| 4. | 1 | 1 | |

Search phases

1. find alignments for cases 1 and 2 (mirror)
2. find *partial* alignments for case 3 (forward)
3. use partial alignments and make full alignments for case 3 and 4 (mirror)

Bowtie

Phased Maq-like Search (cont'd)

Cases with two mismatches allowed:

| | | | |
|----|-----|-----|--|
| 1. | 0 | 0 | |
| 2. | 0 | 1-2 | |
| 3. | 1-2 | 0 | |
| 4. | 1 | 1 | |

Search phases

1. find alignments for cases 1 and 2 (mirror)
2. find *partial* alignments for case 3 (forward)
3. use partial alignments and make full alignments for case 3 and 4 (mirror)

Bowtie

Performance and Index Building

Index building

- ▶ indexer is a stand-alone tool
 - ▶ create index once and re-use for every run
 - ▶ pre-built indices are freely available for download
- ▶ can be configured to trade-off between memory usage and running time
- ▶ implementation uses only one CPU

Parallel performance

- ▶ search can be parallelized
 - ▶ as long as index is available to all
 - ▶ uses pthreads in Bowtie tool

Bowtie

Performance and Index Building

Index building

- ▶ indexer is a stand-alone tool
 - ▶ create index once and re-use for every run
 - ▶ pre-built indices are freely available for download
- ▶ can be configured to trade-off between memory usage and running time
- ▶ implementation uses only one CPU

Parallel performance

- ▶ search can be parallelized
 - ▶ as long as index is available to all
 - ▶ uses pthreads in Bowtie tool

Bowtie

Performance and Index Building

Index building

- ▶ indexer is a stand-alone tool
 - ▶ create index once and re-use for every run
 - ▶ pre-built indices are freely available for download
- ▶ can be configured to trade-off between memory usage and running time
- ▶ **implementation uses only one CPU**

Parallel performance

- ▶ search can be parallelized
 - ▶ as long as index is available to all
 - ▶ uses pthreads in Bowtie tool

Bowtie

Performance and Index Building

Index building

- ▶ indexer is a stand-alone tool
 - ▶ create index once and re-use for every run
 - ▶ pre-built indices are freely available for download
- ▶ can be configured to trade-off between memory usage and running time
- ▶ implementation uses only one CPU

Parallel performance

- ▶ search can be parallelized
 - ▶ as long as index is available to all
 - ▶ uses pthreads in Bowtie tool

Bowtie

Performance in numbers

| Read length | CPU time | Peak memory usage | reads aligned |
|-------------|-------------|-------------------|---------------|
| 36 bp | 6 min 15 s | 1 305 MB | 62.2% |
| 50 bp | 7 min 11 s | 1 310 MB | 67.5% |
| 76 bp | 18 min 58 s | 1 155 MB | 44.9% |

Table: Bowtie v0.9.6, 2 000 000 single reads, human reference genome

SOAP

Short Oligonucleotide Alignment Program

- ▶ Karsten Kristiansen et. al, 2008. University of Southern Denmark, Odense, Denmark, Beijing Genomics Institute, Shenzhen, China [5]
- ▶ basis for SOAP2
- ▶ supports gapped and ungapped read lengths up to 60bp
- ▶ implementation supports multithreading
- ▶ inspired by Maq

SOAP

Short Oligonucleotide Alignment Program

- ▶ Karsten Kristiansen et. al, 2008. University of Southern Denmark, Odense, Denmark, Beijing Genomics Institute, Shenzhen, China [5]
- ▶ basis for SOAP2
- ▶ supports gapped and ungapped read lengths up to 60bp
- ▶ implementation supports multithreading
- ▶ inspired by Maq

SOAP

Short Oligonucleotide Alignment Program

- ▶ Karsten Kristiansen et. al, 2008. University of Southern Denmark, Odense, Denmark, Beijing Genomics Institute, Shenzhen, China [5]
- ▶ basis for SOAP2
- ▶ supports gapped and ungapped read lengths up to 60bp
- ▶ implementation supports multithreading
- ▶ inspired by Maq

SOAP

Short Oligonucleotide Alignment Program

- ▶ Karsten Kristiansen et. al, 2008. University of Southern Denmark, Odense, Denmark, Beijing Genomics Institute, Shenzhen, China [5]
- ▶ basis for SOAP2
- ▶ supports gapped and ungapped read lengths up to 60bp
- ▶ implementation supports multithreading
- ▶ inspired by Maq

SOAP

Short Oligonucleotide Alignment Program

- ▶ Karsten Kristiansen et. al, 2008. University of Southern Denmark, Odense, Denmark, Beijing Genomics Institute, Shenzhen, China [5]
- ▶ basis for SOAP2
- ▶ supports gapped and ungapped read lengths up to 60bp
- ▶ implementation supports multithreading
- ▶ inspired by Maq

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Strategy

- ▶ encode every base as two bits (reference and reads)
- ▶ create seed index table on reference

Matching Strategy

1. create seeds from current read (like in Maq)
 2. find matching seeds
 3. look up positions in reference sequence
 4. align whole read on reference
 - ▶ calculate difference
-
- ▶ too many mismatches \Rightarrow trim read and start over

SOAP

Performance

Very high RAM usage:

$$\frac{L}{3} + (4 \cdot 3 + 8 \cdot 6) \cdot 4^S + (4 + 1) \cdot 3 \cdot \frac{L}{4} + 4 \cdot 2^{24}$$

where

- ▶ L be the reference sequence length [bp]
- ▶ S be the seed length [bp]

Examples

| | sequence length L | seed length S | RAM usage |
|--------------|---------------------|-----------------|-----------|
| yeast | 12 000 | 10 | 200 MB |
| human genome | 3 000 000 000 | 12 | 14 700 MB |

SOAP

Performance

Very high RAM usage:

$$\frac{L}{3} + (4 \cdot 3 + 8 \cdot 6) \cdot 4^S + (4 + 1) \cdot 3 \cdot \frac{L}{4} + 4 \cdot 2^{24}$$

where

- ▶ L be the reference sequence length [bp]
- ▶ S be the seed length [bp]

Examples

| | sequence length L | seed length S | RAM usage |
|--------------|---------------------|-----------------|-----------|
| yeast | 12 000 | 10 | 200 MB |
| human genome | 3 000 000 000 | 12 | 14 700 MB |

SOAP

Performance

Very high RAM usage:

$$\frac{L}{3} + (4 \cdot 3 + 8 \cdot 6) \cdot 4^S + (4 + 1) \cdot 3 \cdot \frac{L}{4} + 4 \cdot 2^{24}$$

where

- ▶ L be the reference sequence length [bp]
- ▶ S be the seed length [bp]

Examples

| | sequence length L | seed length S | RAM usage |
|--------------|---------------------|-----------------|-----------|
| yeast | 12 000 | 10 | 200 MB |
| human genome | 3 000 000 000 | 12 | 14 700 MB |

SOAP2

Improved SOAP

- ▶ supports read lengths up to 1 024 base pairs [6]
 - ▶ designed for read lengths >50 base pairs
- ▶ uses compressed Burrows-Wheeler transform instead of hash table
 - ▶ pre-built (as in Bowtie)
- ▶ basically same strategy as in SOAP
- ▶ performance gain over SOAPv1 (human genome)
 - ▶ memory consumption from 14.7 GB to 5.4 GB
 - ▶ “more than 20 times faster”

SOAP2

Improved SOAP

- ▶ supports read lengths up to 1 024 base pairs [6]
 - ▶ designed for read lengths >50 base pairs
- ▶ uses compressed Burrows-Wheeler transform instead of hash table
 - ▶ pre-built (as in Bowtie)
- ▶ basically same strategy as in SOAP
- ▶ performance gain over SOAPv1 (human genome)
 - ▶ memory consumption from 14.7 GB to 5.4 GB
 - ▶ “more than 20 times faster”

SOAP2

Improved SOAP

- ▶ supports read lengths up to 1 024 base pairs [6]
 - ▶ designed for read lengths >50 base pairs
- ▶ uses compressed Burrows-Wheeler transform instead of hash table
 - ▶ pre-built (as in Bowtie)
- ▶ basically same strategy as in SOAP
- ▶ performance gain over SOAPv1 (human genome)
 - ▶ memory consumption from 14.7 GB to 5.4 GB
 - ▶ “more than 20 times faster”

SOAP2

Improved SOAP

- ▶ supports read lengths up to 1 024 base pairs [6]
 - ▶ designed for read lengths >50 base pairs
- ▶ uses compressed Burrows-Wheeler transform instead of hash table
 - ▶ pre-built (as in Bowtie)
- ▶ basically same strategy as in SOAP
- ▶ performance gain over SOAPv1 (human genome)
 - ▶ memory consumption from 14.7 GB to 5.4 GB
 - ▶ “more than 20 times faster”

SOAP/SOAP2

Parallel Performance

Parallelization possible, if reference index is globally available.

- ▶ read-wise distribution
- ▶ and seed-wise distribution (if multi-matches wanted)

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Feature Matrix

| | Bowtie | SOAP | SOAP2 |
|----------------------------|-------------------|-------------|----------------------|
| max. read length | 1 024 bp | 60 bp | 1 024 bp |
| gaps allowed | no | one, 1-3 bp | one, 1-3 bp |
| paired-end sequencing | yes ¹ | yes | yes |
| uses Phred score? | yes | no | no |
| pre-built index available? | yes | no | no |
| parallel computing | pthreads | threads | threads ² |
| Illumina/SOLiD? | both ³ | Illumina | Illumina |
| License | Artistic | GPL-3.0 | closed |

¹since v0.9.9.1

²since v2.20

³since v0.12

Performance Matrix

| | Bowtie | SOAP |
|---------------|--------------|-----------------|
| RAM usage | 1 138 MB | 13 619 MB |
| CPU time | 4 min 55 sec | 16 h 44 m 3 sec |
| time | 5 min | 18 h 1 min 38 s |
| reads aligned | 55.0 % | 55.1 % |

Table: 2 000 000×36 bp single-ended reads. 2.4 GHz quad-core, 32 GB RAM [Bowtie]

Performance Matrix

| | Bowtie | SOAP |
|---------------|--------------|-----------------|
| RAM usage | 1 138 MB | 13 619 MB |
| CPU time | 4 min 55 sec | 16 h 44 m 3 sec |
| time | 5 min | 18 h 1 min 38 s |
| reads aligned | 55.0 % | 55.1 % |

Table: 2 000 000×36 bp single-ended reads. 2.4 GHz quad-core, 32 GB RAM [Bowtie]

| | Bowtie | SOAP | SOAP2 |
|---------------|--------------|----------------|-------------|
| RAM usage | 2.3 GB | 14.7 GB | 5.4 GB |
| time (single) | 6 min 45 sec | 3h 58 min 48 s | 7 min 58 s |
| time (paired) | - | 5h 20 min 34 s | 13 min 48 s |
| reads aligned | 91.7 % | 93.8 % | 93.6 % |

Table: 1 000 000×44 bp *filtered* reads. 2.0 GHz quad-core, 16 GB RAM [SOAP2]

Cloud Computing

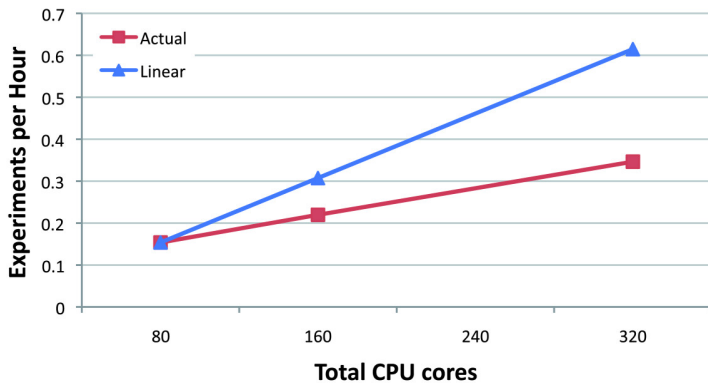
Cherry Picking from Bowtie & SOAP

Crossbow: cloud computing software [3]

- ▶ uses Bowtie for read alignment and SOAPsnp for SNP calling
- ▶ MapReduce application using Apache Hadoop
- ▶ human genome took 2h 53min for 2.66 billion reads on 320-core cluster
 - ▶ 85 USD on an Amazon EC2 Cloud

Cloud Computing

Crossbow Speedup



References I

Thank you for listening!



M. Burrows and D. Wheeler.

A block-sorting lossless data compression algorithm.

Technical report, Digital SRC Research Report, 1994.



P. Ferragina and G. Manzini.

Opportunistic data structures with applications.

In *FOCS*, pages 390–398, 2000.



B. Langmead, M. Schatz, J. Lin, M. Pop, and S. Salzberg.

Searching for SNPs with cloud computing.

Genome Biology, 10(11):R134, 2009.

References II



B. Langmead, C. Trapnell, M. Pop, and S. Salzberg.

Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.

Genome Biology, 10(3):R25, 2009.



R. Li, Y. Li, K. Kristiansen, and J. Wang.

SOAP: short oligonucleotide alignment program.

Bioinformatics, 24(5):713–714, 2008.



R. Li, C. Yu, Y. Li, T. W. Lam, S.-M. Yiu, K. Kristiansen, and J. Wang.

SOAP2: an improved ultrafast tool for short read alignment.

Bioinformatics, 25(15):1966–1967, 2009.