Bowtie: A Highly Scalable Tool for Post-Genomic Datasets

Ben Langmead (langmead@cs.umd.edu)
Work with: Cole Trapnell, Mihai Pop, Steven Salzberg

NCBI Seminar
November 10, 2008
Supply

- High-throughput sequencing produces short DNA sequences ("reads") in huge volumes at low cost
- Costs are down, adoption is up, next-next generation is coming soon

Illumina  
ABI  
454
Demand

- Researchers have no problem putting this data to work

1000 Genomes  Global Ocean Survey  Human Microbiome

- Future growth may be multiplicative
  - Spatial resolution: tissues, individuals, geography
  - Temporal resolution: circadian, seasonal, lifetimes
Short Read Applications

- **Genotyping**
  - Goal: identify variations
  
  ```
  ...CCATAG TATGCGCCC CGGAATTT CGGTATAC
  ...CCAT CTATATGCG TCGGAATTT CGGTATAC
  ...CCAT GGCTATATG CTATCGGAAA GCGGTATA
  ...CCA AGGCTATAT CCTATCGGA TTGCGGTA C...
  ...CCA AGGCTATAT GCCCTATCG TTTGCGGT C...
  ...CC AGGCTATAT GCCCTATCG AATTTCG ATAC...
  ...CC TAGGCTATA GCGCCCTA AATTTCG GTATAC...
  ...CCATAGGCTATATGCGGCCCCTATCGGCAAATTTCGCGGTATAC...
  ```

- **RNA-seq, ChIP-seq, Methyl-seq**
  - Goal: classify, measure significant peaks
  
  ```
  GAAATTTGC
  GGAATTTTGC
  CGGAAATTT
  CGGAAATTT
  TCGGAAATTT
  CTATCGGAAA
  CCTATCGGA TTTGCGGT
  GCCCTATCG AAATTTCG
  GCCCTATCG AAATTTCG ATAC...
  ```

```
Finding the alignments is typically the performance bottleneck.
Short Read Alignment

- Given a reference and a set of reads, report at least one “good” local alignment for each read if one exists
  - Approximate answer to: where in genome did read originate?

- What is “good”? For now, we concentrate on:
  - Fewer mismatches is better
  - Failing to align a low-quality base is better than failing to align a high-quality base
Indexing

- Genomes and reads are too large for direct approaches like dynamic programming

- **Indexing** is required

- Choice of index is key to performance

[Diagram of Suffix tree and Suffix array showing nodes and edges]

**Suffix tree**

**Suffix array**

**Seed hash tables**

Many variants, incl. spaced seeds
Indexing

- Genome indices can be big. For human:
  1. Require big-memory machine
  2. Use secondary storage
  3. Build new index each run
  4. Subindex and do multiple passes

- Large indices necessitate painful compromises

  > 35 GBs
  > 12 GBs
  > 12 GBs
Burrows-Wheeler Transform

- Reversible permutation used originally in compression

\[ a c a a c g$ \xrightarrow{T} a c a a c g \xrightarrow{BWT(T)} g c $ a a a c \]

- Once BWT(T) is built, all else shown here is discarded
  - Matrix will be shown for illustration only

Burrows-Wheeler Transform

- Property that makes BWT(T) reversible is “LF Mapping”
  - \(i^{th}\) occurrence of a character in Last column is same \textit{text} occurrence as the \(i^{th}\) occurrence in First column
Burrows-Wheeler Transform

- To recreate $T$ from $BWT(T)$, repeatedly apply rule:
  $$T = BWT[LF(i)] + T; \ i = LF(i)$$
  - Where $LF(i)$ maps row $i$ to row whose first character corresponds to $i$’s last per LF Mapping

- Could be called “unpermute” or “walk-left” algorithm
Ferragina & Manzini propose “FM Index” based on BWT

Observed:
– LF Mapping also allows exact matching within \( T \)
– \( \text{LF}(i) \) can be made fast with checkpointing
– ...and more (see FOCS paper)


Exact Matching with FM Index

- To match Q in T using BWT(T), repeatedly apply rule:
  \[ \text{top} = \text{LF}(\text{top}, \text{qc}); \ \text{bot} = \text{LF}(\text{bot}, \text{qc}) \]
  - Where \( \text{qc} \) is the next character in Q (right-to-left) and \( \text{LF}(i, \text{qc}) \) maps row i to the row whose first character corresponds to i’s last character as if it were \( \text{qc} \)
Exact Matching with FM Index

- In progressive rounds, \textbf{top} \& \textbf{bot} delimit the range of rows beginning with progressively longer suffixes of $Q$. 
Exact Matching with FM Index

• If range becomes empty (top = bot) the query suffix (and therefore the query) does not occur in the text
Once we know a row contains a legal alignment, how do we determine its position in the reference?
Rows to Reference Positions

- Naïve solution 1: Use “walk-left” to walk back to the beginning of the text; number of steps = offset of hit

2 steps, so hit offset = 2

- Linear in length of text in general – too slow
Naïve solution 2: Keep whole suffix array in memory. Finding reference position is a lookup in the array.

- Suffix array is \( \sim 12 \) gigabytes for human – too big
Rows to Reference Positions

- Hybrid solution: Store *sample* of suffix array; “walk left” to next sampled (“marked”) row to the left
  - Due to Ferragina and Manzini

- Bowtie marks every 32\textsuperscript{nd} row by default (configurable)
Put It All Together

- Algorithm concludes: “aac” occurs at offset 2 in “acaacg”
Checkpointing in FM Index

- \( \text{LF}(i, \text{qc}) \) must determine the *rank* of \( \text{qc} \) in row \( i \)
- Naïve way: count occurrences of \( \text{qc} \) in all previous rows
  - This \( \text{LF}(i, \text{qc}) \) is linear in length of text – too slow

Scanned by naïve rank calculation
Checkpointing in FM Index

- Solution: pre-calculate cumulative counts for A/C/G/T up to periodic **checkpoints** in BWT

- \( LF(i, qc) \) is now constant-time
  (if space between checkpoints is considered constant)
FM Index is Small

- Entire FM Index on DNA reference consists of:
  - BWT (same size as T)
  - Checkpoints (~15% size of T)
  - SA sample (~50% size of T)

- Total: ~1.65x the size of T

Assuming 2-bit-per-base encoding and no compression, as in Bowtie
Assuming a 16-byte checkpoint every 448 characters, as in Bowtie
Assuming Bowtie defaults for suffix-array sampling rate, etc
FM Index in Bioinformatics

• Oligomer counting

• Whole-genome alignment

• Smith-Waterman alignment to large reference
Short Read Alignment

- FM Index finds exact sequence matches quickly in small memory, but short read alignment demands more:
  - Allowances for mismatches
  - Consideration of quality values
- Lam et al try index-assisted Smith-Waterman
  - Slower than BLAST
- We tried index-assisted “seed-and-extend”
  - Competitive with other aligners, but not much faster
- Bowtie’s solution: backtracking quality-aware search
Backtracking

• Consider an attempt to find $Q = \text{“agc”}$ in $T = \text{“acaacg”}$:

$\text{\$ a c a a c g}$ $\text{\$ a c a a c g}$
$\text{a a c g \$ a c}$ $\text{a a c g \$ a c}$
$\text{a c a a c g \$}$ $\text{a c a a c g \$}$
$\text{a c g \$ a c a}$ $\text{a c g \$ a c a}$
$\text{c a a c g \$ a}$ $\text{c a a c g \$ a}$
$\text{c g \$ a c a a}$ $\text{c g \$ a c a a}$
$\text{g \$ a c a a c}$ $\text{g \$ a c a a c}$

“gc” does not occur in the text

• Instead of giving up, try to “backtrack” to a previous position and try a different base
Backtracking

- Backtracking attempt for Q = “agc”, T = “acaacg”:

$ a c a a c g$
$a a c g $ a c$
$a c a a c g $
$a c g $ a c a$
$c a a c g $ a$
$c g $ a c a a$
$g $ a c a a c$

“g”

“c”

Substitution

“a”

“a”

“gc” does not occur in the text

Found this alignment:

acaacg

agc
Backtracking

- May not be so lucky

```plaintext
Found this alignment (eventually):

```

acaacgc
```

| acaacgc
| aacgc
| acgc
| caacgc
| caacgc
| gc
```
Backtracking

- Relevant alignments may lie along multiple paths
  - E.g., Q = “aaa”, T = “acaacg”
Backtracking

- Bowtie’s \(-v\) \(<\text{int}>\) option allows alignments with up to \(<\text{int}>\) mismatches
  - Regardless of quality values
  - Max mismatches allowed: 3
  - Equivalent to SOAP’s* \(-v\) option

Qualities

• When backtracking is necessary, Bowtie will backtrack to leftmost just-visited position with minimal quality

<table>
<thead>
<tr>
<th>Sequence:</th>
<th>G</th>
<th>C</th>
<th>C</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phred Quals:</td>
<td>40</td>
<td>40</td>
<td>35</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>15</td>
<td>15</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

(higher number = higher confidence)

• Greedy, depth-first, not optimal, but simple
Qualities

- Bowtie supports a Maq*-like alignment policy
  - \( \leq N \) mismatches allowed in first \( L \) bases on left end
  - Sum of mismatch qualities may not exceed \( E \)
  - \( N, L \) and \( E \) configured with \(-n, -l, -e\)
  - E.g.:

<table>
<thead>
<tr>
<th>G</th>
<th>C</th>
<th>C</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>40</td>
<td>35</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>15</td>
<td>15</td>
<td>40</td>
<td>25</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

- Maq-like is Bowtie’s default mode (\( N=2, L=28, E=70 \))

Implementation

- Free, Open Source under Artistic License
- C++
- Uses SeqAn* library (http://www.seqan.de)
- Uses POSIX threads to exploit parallelism
- `bowtie-build` is the indexer
- `bowtie` is the aligner
- `bowtie-convert` converts Bowtie’s alignment output format to Maq’s `.map` format
  - Users may leverage tools in the Maq suite, e.g., `maq assemble`, `maq cns2snp`
  - Uses code from Maq

Bowtie is an ultrafast, memory-efficient short read aligner. It aligns short DNA sequences (reads) to the human genome at a rate of 25 million reads per hour on a typical workstation with 2 gigabytes of memory. Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: 1.3 GB for the human genome. It supports alignment policies equivalent to Maq and SOAP but is much faster: about 35x faster than Maq and over 350x faster than SOAP when aligning to the human genome.

0.9.7 release - 11/8/08
- Added new reporting option -m <int> which suppresses all alignments for a particular read if more than <int> reportable alignments exist for it.
- Threads now buffer all alignments for a particular read/phase then output all alignments in one critical section. This guarantees that all alignments for a given read/phase appear in one consecutive block of the output, even when multiple threads are operating in parallel.
- Separated the quality-convension and parsing aspects of the old --solexa-quals argument into separate arguments: --solexa-quals (quality conversion) and --integer-quals (parsing).
- bowtie-convert now handles the new (post-0.7.0) Maq alignment format. The new format allows Maq tools to handle reads up to 127 bases, whereas the old format was limited to 63 bases. Added a --o option to opt for the old Maq format.
- New --refout argument sends alignments to a set of files named refxxxxx.map, where xxxx is the 0-padded index of the reference sequence aligned to. Useful for dealing with large datasets aligned to, e.g., the assembled human genome.
- Improved tutorial to use a simple simulated read set (included) to do SNP calls with Maq.
- Added --nota option to bowtie-build
- Fixed make h_sapiens_asn.sh script to include mitochondrial DNA.

TopHat released - 11/8/08
- Cole Trapnell has completed the initial release of TopHat, a fast splice junction mapper for RNA-Seq reads. TopHat aligns RNA-Seq reads to mammalian-sized genomes using Bowtie, and then analyzes the mapping results to identify splice junctions between exons.
Indexing Performance

• Bowtie employs a indexing algorithm* that can trade flexibly between memory usage and running time
• For human (NCBI 36.3) on 2.4 GHz AMD Opteron:

<table>
<thead>
<tr>
<th>Physical memory Target</th>
<th>Actual peak memory footprint</th>
<th>Wall clock time</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 GB</td>
<td>14.4 GB</td>
<td>4h:36m</td>
</tr>
<tr>
<td>8 GB</td>
<td>5.84 GB</td>
<td>5h:05m</td>
</tr>
<tr>
<td>4 GB</td>
<td>3.39 GB</td>
<td>7h:40m</td>
</tr>
<tr>
<td>2 GB</td>
<td>1.39 GB</td>
<td>21h:30m</td>
</tr>
</tbody>
</table>

Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td>59.8x</td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td>107x</td>
<td>74.7</td>
</tr>
</tbody>
</table>

- PC: 2.4 GHz Intel Core 2, 2 GB RAM
- Server: 2.4 GHz AMD Opteron, 32 GB RAM
- Bowtie v0.9.6, Maq v0.6.6, SOAP v1.10
- SOAP not run on PC due to memory constraints
- Reads: FASTQ 8.84 M reads from 1000 Genomes (Acc: SRR001115)
- Reference: Human (NCBI 36.3, contigs)
## Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td><strong>33.8 M</strong></td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td><strong>351x</strong></td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td><strong>29.5 M</strong></td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td><strong>59.8x</strong></td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td><strong>28.8 M</strong></td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td><strong>107x</strong></td>
<td>74.7</td>
</tr>
</tbody>
</table>

- Bowtie delivers about 30 million alignments per CPU hour
Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td>59.8x</td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td>107x</td>
<td>74.7</td>
</tr>
</tbody>
</table>

- Disparity in reads aligned between Bowtie (67.4%) and SOAP (67.3%) is slight
Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td>1,353 MB</td>
<td>-</td>
<td><strong>71.9</strong></td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td>59.8x</td>
<td><strong>74.7</strong></td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td><strong>71.9</strong></td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td>107x</td>
<td><strong>74.7</strong></td>
</tr>
</tbody>
</table>

- Disparity in reads aligned between Bowtie (71.9%) and Maq (74.7%) is more substantial (2.8%)
  - Mostly because Maq –n 2 reports some, but not all, alignments with 3 mismatches in first 28 bases
  - Fraction (<5%) of disparity is due to Bowtie’s backtracking limit (a heuristic not discussed here)
## Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td><strong>1,149 MB</strong></td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td><strong>13,619 MB</strong></td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td><strong>1,353 MB</strong></td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td><strong>804 MB</strong></td>
<td>59.8x</td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td><strong>1,353 MB</strong></td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td><strong>804 MB</strong></td>
<td>107x</td>
<td>74.7</td>
</tr>
</tbody>
</table>

- **Bowtie and Maq** have memory footprints compatible with a typical workstation with 2 GB of RAM
  - Maq builds non-reusable spaced-seed index on reads; recommends segmenting reads into chunks of 2M (which we did)
- **SOAP** requires a computer with >13 GB of RAM
  - SOAP builds non-reusable spaced-seed index on genome
### Comparison to Maq w/ Poly-A Filter

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bowtie (PC)</strong></td>
<td>16m:39s</td>
<td>17m:47s</td>
<td>29.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>74.9</td>
</tr>
<tr>
<td><strong>Maq (PC)</strong></td>
<td>11h:15m:58s</td>
<td>11h:22m:02s</td>
<td>0.78 M</td>
<td>804 MB</td>
<td>38.4x</td>
<td>78.0</td>
</tr>
<tr>
<td><strong>Bowtie (server)</strong></td>
<td>18m:20s</td>
<td>18m:46s</td>
<td>28.8 M</td>
<td>1,352 MB</td>
<td>-</td>
<td>74.9</td>
</tr>
<tr>
<td><strong>Maq (server)</strong></td>
<td>18h:49m:07s</td>
<td>18h:50m:16s</td>
<td>0.47 M</td>
<td>804 MB</td>
<td>60.2x</td>
<td>78.0</td>
</tr>
</tbody>
</table>

- Maq documentation: reads with “poly-A artifacts” impair Maq’s performance
- Re-ran previous experiment after running Maq’s “catfilter” to eliminate 438K poly-A reads
- Maq makes up some ground, but Bowtie still >35x faster
- Similar disparity in reads aligned, for same reasons
Multithreaded Scaling

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie, 1 thread (server)</td>
<td>18m:19s</td>
<td>18m:46s</td>
<td>28.3 M</td>
<td>1,353 MB</td>
<td>-</td>
</tr>
<tr>
<td>Bowtie, 2 threads (server)</td>
<td>20m:34s</td>
<td>10m:35s</td>
<td>50.1 M</td>
<td>1,363 MB</td>
<td>1.77x</td>
</tr>
<tr>
<td>Bowtie, 4 threads (server)</td>
<td>23m:09s</td>
<td>6m:01s</td>
<td>88.1 M</td>
<td>1,384 MB</td>
<td>3.12x</td>
</tr>
</tbody>
</table>

• Bowtie uses POSIX threads to exploit multi-processor computers
  – Reads are distributed across parallel threads
  – Threads synchronize when fetching reads, outputting results, etc.
  – Index is shared by all threads, so footprint does not increase substantially as # threads increases

• Table shows performance results for Bowtie v0.9.6 on 4-core Server with 1, 2, 4 threads
1000 Genomes Genotyping

- Bowtie aligns all 1000-Genomes (Build 2) reads for human subject NA12892 on a 2.4 Ghz Core 2 workstation with 4 GB of RAM with 4 parallel threads:
  - 14.3x coverage, 935 M reads, 42.9 Gbases
  - Running time: 14 hrs – 1 overnight
Future Work

- Paired-end alignment
- Finding alignments with insertions and deletions
- ABI color-space support
TopHat: Bowtie for RNA-seq

- TopHat is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads using Bowtie, and then analyzes the mapping results to identify splice junctions between exons.
  - Contact: Cole Trapnell (cole@cs.umd.edu)
  - http://tophat.cbcb.umd.edu
Work With

Cole Trapnell  Steven Salzberg  Mihai Pop

Center for Bioinformatics & Computational Biology
Extra Slides
Bowtie Usage

Usage: bowtie [options]* <ebwt_base> <query_in> [hit_outfile]

<ebwt_base>        ebwt filename minus trailing .1.ebwt/.2.ebwt
<query_in>         comma-separated list of files containing query reads
                  (or the sequences themselves, if -c is specified)
<hit_outfile>      file to write hits to (default: stdout)

Options:
-q                 query input files are FASTQ .fq/.fastq (default)
-f                 query input files are (multi-)FASTA .fa/.mfa
-c                 query sequences given on command line (as <query_in>)
-e/--maqerr <int>  max sum of mismatch quals (rounds like maq; default: 70)
-l/--seedlen <int> seed length (default: 28)
-n/--seedmms <int> max mismatches in seed (0, 1 or 2, default: 2)
-v <int>           report end-to-end hits w/ <=v mismatches; ignore qualities
-5/--trim5 <int>   trim <int> bases from 5' (left) end of reads
-3/--trim3 <int>   trim <int> bases from 3' (right) end of reads
-u/--qupto <int>   stop after the first <int> reads
-t/--time          print wall-clock time taken by search phases
--solexa-quals     convert FASTQ qualities from solexa-scaled to phred
--concise          write hits in a concise format
--maxns <int>      skip reads w/ >n no-confidence bases (default: no limit)
-o/--offrate <int> override offrate of Ebwt; must be >= value in index
--seed <int>       seed for random number generator
--verbose          verbose output (for debugging)
--version          print version information and quit
Bowtie Indexer Usage

Usage: bowtie-build [options]* <reference_in> <ebwt_outfile_base>

   reference_in            comma-separated list of files with ref sequences
   ebwt_outfile_base       write Ebwt data to files with this dir/basename

Options:
- reference files are Fasta (default)
  -f
- reference sequences given on cmd line (as <seq_in>)
  -c
- max bucket sz for blockwise suffix-array builder
  --bmax <int>
- max bucket sz as multiple of sqrt(ref len)
  --bmaxmultsqrt <int>
- max bucket sz as divisor of ref len
  --bmaxdivn <int>
- diff-cover period for blockwise (default: 1024)
  --dcv <int>
- disable difference cover (blockwise is quadratic)
  --nodc
- SA index is kept every 2^offRate BWT chars
  -o/--offrate <int>
- # of characters in initial lookup table key
  -t/--ftabchars <int>
- endianness (default: little, this host: little)
  --big --little
- seed for random number generator
  --seed <int>
- truncate reference at prefix of <int> bases
  --cutoff <int>
- verbose output (for debugging)
  -q/--quiet
- print detailed description of tool and its options
  -h/--help
- print version information and quit
  --version

--version
## Reporting

- **-k <int>** Report up to <int> valid alignments per read (default: 1). Validity of alignments is determined by the alignment policy (combined effects of -n, -v, -l, and -e). If many alignments are reported, they may be subject to stratification; see --best, --nostrata. Bowtie is designed to be very fast for small -k but **BOWTIE CAN BECOME VERY SLOW AS -k INCREASES.**

- **-a/--all** Report all valid alignments per read (default: off). Validity of alignments is determined by the alignment policy (combined effects of -n, -v, -l, and -e). Reported alignments may be subject to stratification; see --best, --nostrata. Bowtie is designed to be very fast for small -k; **BOWTIE CAN BECOME VERY SLOW IF -a/--all IS SPECIFIED.**

- **--best** Reported alignments must belong to the best possible alignment "stratum" (default: off). A stratum is a category defined by the number of mismatches present in the alignment (for -n, the number of mismatches present in the seed region of the alignment). E.g., if --best is not specified, Bowtie may sometimes report an alignment with 2 mismatches in the seed even though there exists an unreported alignment with 1 mismatch in the seed. **bowtie IS ABOUT 3-5 TIMES SLOWER WHEN --best IS SPECIFIED.**

- **--nostrata** If many valid alignments exist and are reportable (according to the --best and -k options) and they fall into various alignment "strata", report all of them. By default, Bowtie only reports those alignments that fall into the best stratum, i.e., the one with fewest mismatches. **BOWTIE CAN BECOME VERY SLOW WHEN --nostrata IS COMBINED WITH -k OR -a.**
Excessive Backtracking

- Bowtie only backtracks if it can make progress, i.e., if $\text{top} \neq \text{bot}$ after the backtrack
  - Rightmost positions are likeliest targets because shorter suffixes are likeliest to occur “by chance”

G C C A T A C G G A T T A G C C

- When >1 mismatch is allowed, such backtracks can easily dominate running time and make search slow
Excessive Backtracking

- Solution: Double indexing

- We’ve considered matching from right to left, but what if left-to-right were possible too?

\[
\begin{array}{cccccccccc}
\end{array}
\]

Longer suffixes, less likely targets

Shorter suffixes, more likely targets

\[
\begin{array}{cccccccccc}
\end{array}
\]

Shorter prefixes, more likely targets

Longer prefixes, less likely targets
Excessive Backtracking

- Suggests a multi-stage scheme that minimizes excessive backtracking in reddest regions
  - Workflow for up to 1-mismatch that matches in both directions & disallows backtracks in reddest regions:
Excessive Backtracking

- Minimizes backtracks by disallowing backtracks in reddest regions
- Maintains full sensitivity by matching in both directions
Excessive Backtracking

- But how to match left-to-right?
- Double indexing:
  - Reverse read and use “mirror index”: index for reference with sequence reversed

Forward Index

Mirror Index

No backtracks allowed
“True understanding of how genes function requires knowledge of their expression patterns, their impact on all other genes and their effects on DNA structure and modifications. These data will have to be obtained across large numbers of cell types, individuals, environments and time points.”

Wanted: Scalable Algorithms

“...the overwhelming amounts of data being produced are the equivalent of taking a drink from a fire hose...”

“...grant-awarding bodies should start focusing on the back-end bioinformatics as much as the sequencing technology itself. And as the bioinformatics bottleneck threatens to limit instrument sales, manufacturers as a group have a massive incentive to unblock it.”

- Editorial, Nature Biotechnology 26, 1099 (Oct 2008)
PubMed was “searched in two-year increments for key words and the number of hits plotted over time.”

**From the following article**

*What would you do if you could sequence everything?*

Avak Kahvejian, John Quackenbush & John F Thompson
doi:10.1038/nbt1494