Massively parallel read mapping on graphics cards

Johannes Köster

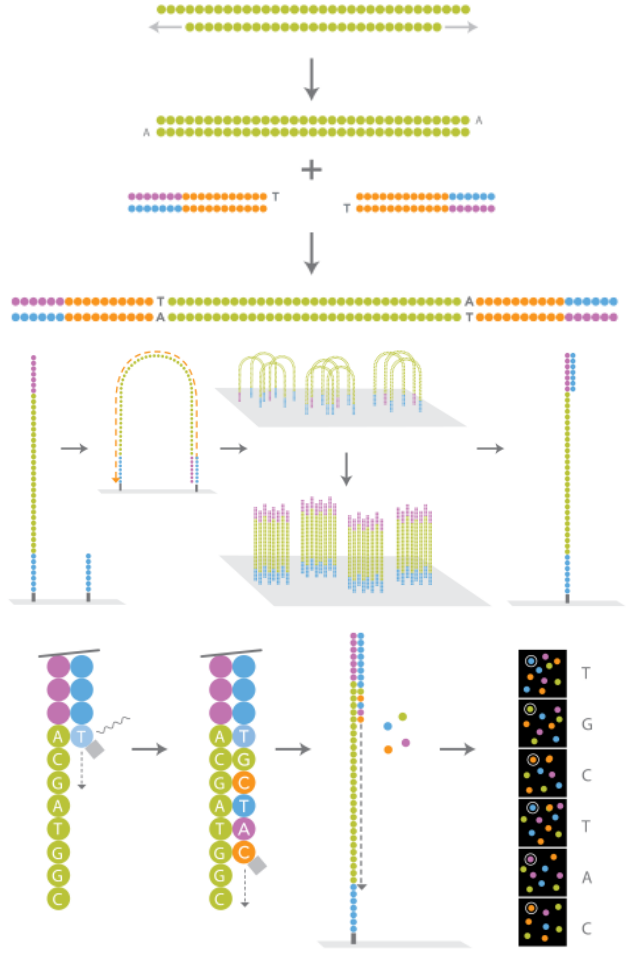
May 15, 2014

Outline

Outline

Next-Generation-Sequencing

1. Chop DNA/RNA into small fragments.
2. Ligate adapters to both ends.
3. Spread fragment solution across a flowcell with beads.
4. Amplify fragments into clusters (PCR).
5. Sequence fragments by adding fluorescent complementary bases ▸ reads.

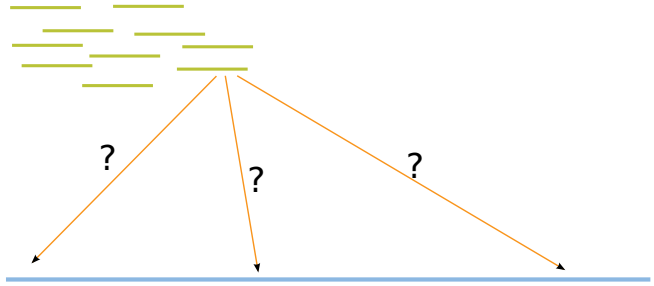


Illumina, 2013

Outline

Read Mapping

For each read...
find position in the known reference genome.



- A DNA sequence is a word over $\Sigma = \{A, C, G, T\}$.
- string matching, but with error tolerance

Read Mapping

For each read...
find position(s) with optimal alignment(s) to either strand of the reference:

ACTGTGGACTATCAATGGAC
GGTACTGT CTATCTATGGACCGTTAG

- Smith Waterman Algorithm

Too slow, therefore heuristics to find anchor positions:

- suffixarray/Burrows-Wheeler-Transformation (BWA, bowtie2)
- q-gram indices (RazerS3)

Read mapping on GPUs

Challenges:

- limited and slow memory ↯ q-gram index
- branching interrupts parallelism ↯ BWT

Idea:

- Use a special q-gram index with small memory footprint.
- Use parallelism to hide memory latency.
- Export branching into bitvector operations.
- ▸ PEANUT – the ParallEl AligNment UTility

Outline

Algorithm

Main steps:

- Filtration
  find potential hits between reads and reference
  using a special q-gram index
- Validation
  validate hits
  using a bit-parallel alignment algorithm

Algorithm

Main steps:

- **Filtration**
  find potential hits between reads and reference
  using a special q-gram index
- Validation
  validate hits
  using a bit-parallel alignment algorithm

Q-Gram Index

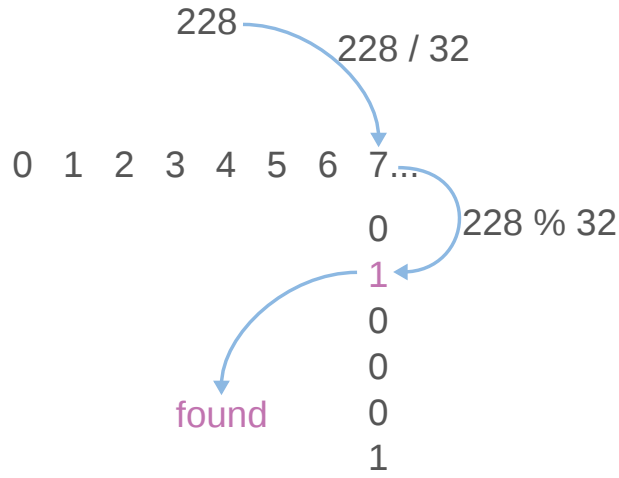
For a given DNA sequence T:

- consider q-grams (substrings of length q)
  GGTACTGACGTTCTATGGACCGTTAG
- encode them as integers
  ACGT $= 11\ 10\ 01\ 00 = 228$
- array P with concatenation of q-gram positions
- array Q with address in P for each q-gram
- size $4^q + |T|$

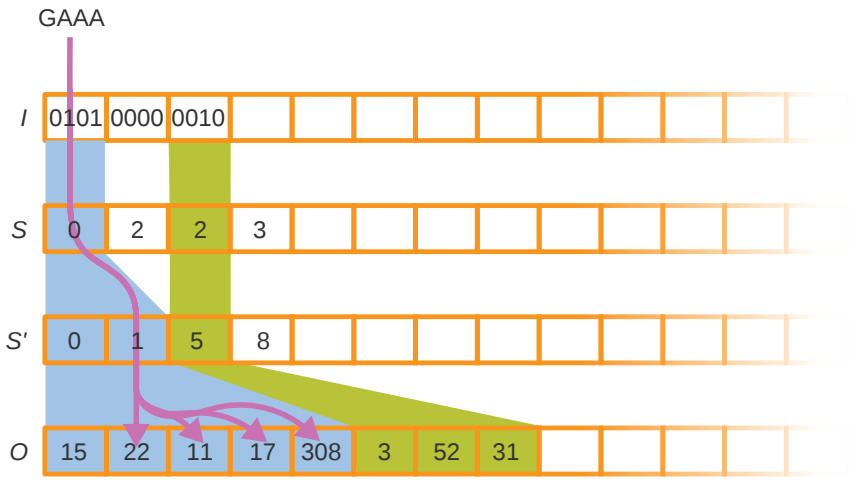
$$P[Q[228]] \ldots P[Q[229]]$$

Q-Group Index

- assign each q-gram to a q-group $\lfloor g/w \rfloor$
- store occurence of q-gram in a bit-vector
- two address arrays guide from q-group to positions of the q-gram in the text
- size $2/w \cdot 4^q + \min\{4^q, |T|\} + |T|$



Q-Group Index



less memory, because we consider only...

- q-groups at the top level
- occuring q-grams at the bottom

calculate adress ranges in parallel by

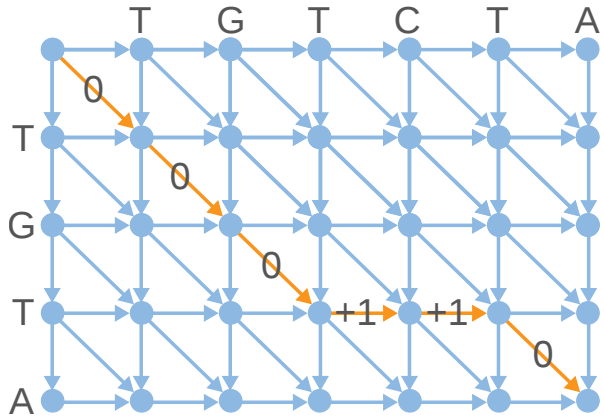
- population counts
- prefix-sums

Algorithm

Main steps:

- Filtration
  find potential hits between reads and reference
  using a special q-gram index
- **Validation**
  validate hits
  using a bit-parallel alignment algorithm

Validation

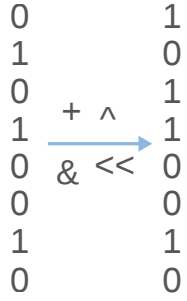


Observations:

- calculating column j needs only column $j-1$
- each transition changes edit distance by at most 1

Myers bit-parallel algorithm[1]:

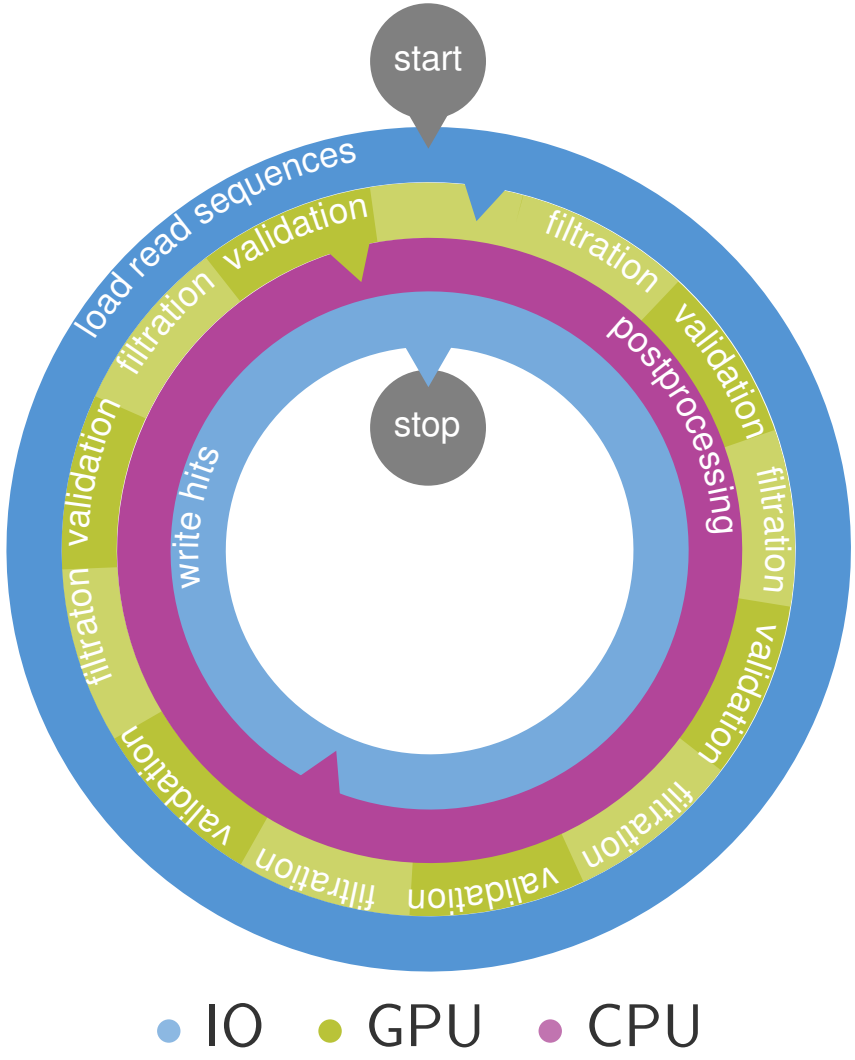
- process graph column-wise
- maintain distance deltas in bitvectors



[1] Myers, 1999. J. ACM 46.

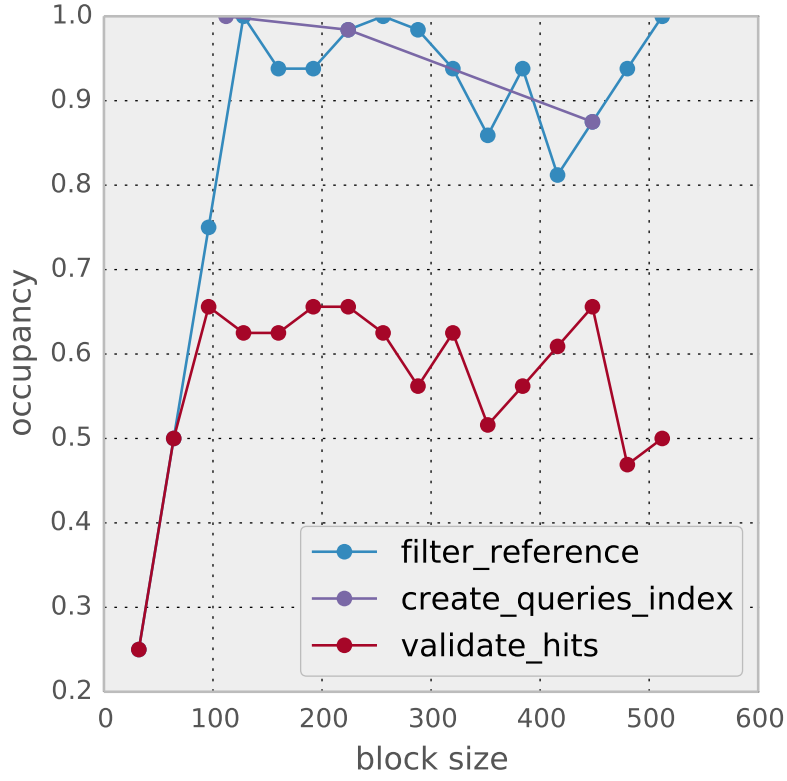
Workflow

- load reads into buffer
- build q-group index of reads
- filtration of hits
- validation of hits
- postprocessing
- writing



Outline

Results


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
occupancy
0
100
200
300
400
500
600
block size
filter_reference
create_queries_index
validate_hits

Sensitivity

- assessed with Rabema[2] benchmark on S. cerevisiae genome
- 100% for reads with error rate less than 7%
- 99.77% for error rates up to 10%
- 98.97% for error rates up to 20%

[2] Holtgrewe et al. 2011. BMC Bioinformatics

Performance

Output types:

- **all** all alignments of a read
- **best** one of the best alignments
- **best-stratum** all best alignments

5 million simulated human reads:

mapper	type	time [min:sec]	sens. [%]
PEANUT	best-stratum	**1:55**	**98.62**
BWA-MEM	best	3:16	96.99
Bowtie 2	best	5:21	96.85
PEANUT	all	**18:29**	98.74
RazerS 3	all	199:55	**98.83**

Intel Core i7, 16GB RAM
NVIDIA Geforce 780, 3GB RAM

Performance

5 million real human exome reads:

mapper	type	time [min:sec]
PEANUT	best-stratum	**1:33**
BWA-MEM	best	1:58
Bowtie 2	best	3:12
PEANUT	all	**10:52**
RazerS 3	all	89:38

Intel Core i7, 16GB RAM
NVIDIA Geforce 780, 3GB RAM

Performance

10 million human exome paired end reads:

mapper	type	time [min:sec]
PEANUT	best-stratum	**3:08**
BWA-MEM	best	4:44
Bowtie 2	best	8:18
PEANUT	all	**21:54**
RazerS 3	all	150:59

Intel Core i7, 16GB RAM
NVIDIA Geforce 780, 3GB RAM

Summary

PEANUT is a GPU based read mapper that outperforms other state-of-the-art mappers in terms of

- sensitivity
- speed

by introducing the q-group index with small memory footprint and exploiting

- bit-vector operations
- prefix sums
- population counts

`http://peanut.readthedocs.org`