SEASHORE / SARUMAN

Short Read Matching using GPU Programming

Tobias Jakobi

Center for Biotechnology (CeBiTec)
Bioinformatics Resource Facility (BRF)
Bielefeld University

28.04.2010

Bielefeld University
Center for Biotechnology (CeBiTec)

BRF

Outline

Short Read Matching

- New sequencing technologies with short reads
  - Solexa, SOLiD
- Reads of length ~35bp
  - Solexa up to 100bp by now
- Mainly used for resequencing
  - Matching reads to a reference genome

”History“

- Bowtie, MAQ, BWA, SOAP2 not yet available
- Normal alignment algorithms not feasable
  - BLAST: Does not work for short reads
  - SSAHA, agrep: No mismatch positions or alignments returned
  - ELAND: Limited to 32bp read length
  - SWIFT: Showed errors for 25bp reads
- Decision to implement own alignment algorithm

SEASHORE

SEmiglobal **A**lignment of **SHO**rt **RE**ads

- Developed by Jochen Blom
- Create qgram index of reference genome
- Use index to estimate auspicious alignment positions
- Modified Needleman-Wunsch alignment
- Parallel calculation on the compute cluster

Read segmentation

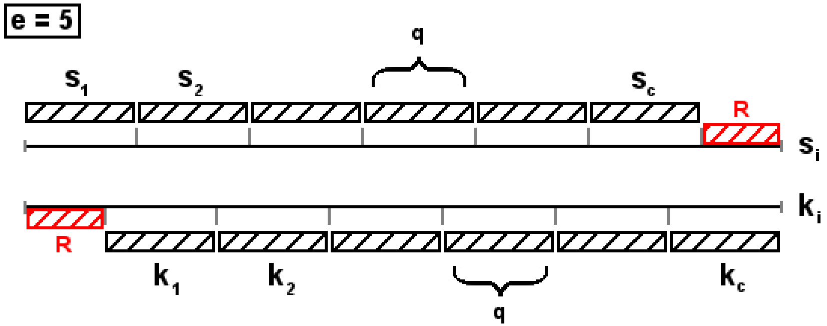


Figure: The two sets of segments S_i and k_i. The two sets are shifted by the distance of R.

Filter strategy

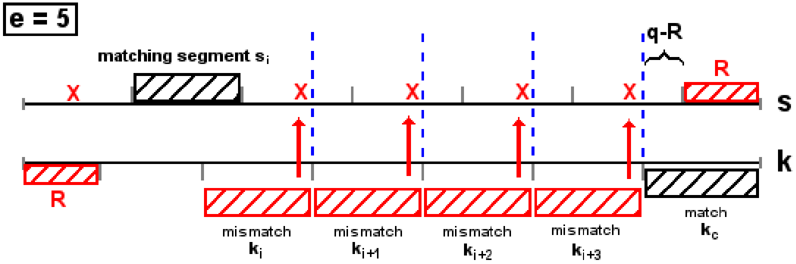


Figure: Try to match all segments of *k* iteratively until a match is found or k_c is reached.

Stats

- Algorithm is exact
    - All possible hits are found
- Reasonable fast
    - Perl implementation
    - 7 mio. reads mapped to bacterial genome
    - 1h on 100 CPUs

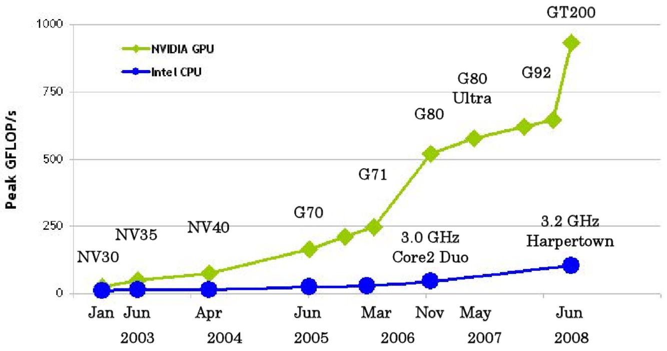
CUDA

- Time consuming step: Alignment
- Alignments are small $(m \times (m + 2e))$
- Can be computed parallel
- Huge amount of small jobs $\rightarrow$ CUDA

CUDA - Compute Unified Device Architecture

- GPU computing power today exceeds 1 TFlops (GTX480, 1,35 TFlops SP / 168 GFlops DP, 1536MB)
  - Reasonable prices for those cards: 500 EUR
  - Even mainstream cards of normal family PCs sufficent
- Use this power to solve computationally intensive problems
- NVIDIA released CUDA API in 2006
  - 'C for CUDA' (C with NVIDIA extensions)
  - Multi platform (32/64Bit): Windows, Linux, Mac OS X

Computing power of recent graphics cards



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

CUDA facts

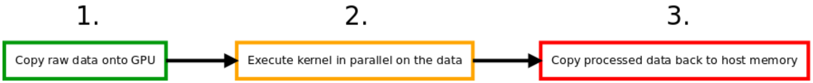
- ”Many core“ architecture of GPUs
  - Optimized hardware for processing massive amounts of data in parallel
  - Up to 512 (stream-)processors per card (varying)
  - Grouped to multiprocessors with shared memory (8-16kb)
- Processor features
  - Full support for integer and bitwise operations
  - Double precision operations
  - Thousands of threads per core

How to use CUDA?

- Requirement:
  - Problem should be dividable ("divide & conquer")
  - → each thread solves a small part of the problem
- Kernel:
  - C-function ("device code") executed on the GPU hundreds of times in parallel with different data
  - Called from the "host code" e.g. C/C++
  - Should be quite simple and relatively easy to compute

General CUDA processing flow

1. Copy input data from host memory to GPU memory
2. One or more kernels are executed on the input data
3. Copy output data back to the host memory


1.
Copy raw data onto GPU
2.
Execute kernel in parallel on the data
3.
Copy processed data back to host memory

SARUMAN runs in two phases

SARUMAN – **S**emiglobal **A**lignment of Short **R**eads **u**sing CUDA and Needle**man**-Wunsch

- Phase one: executed on the host
  - Reads data files (reference genome & reads)
  - Creates q-gram index
  - Runs the SEASHORE matching algorithm
  - Does I/O operations
- Phase two: executed on the GPU
  - Computes the edit-distance of genome and read
  - Does backtracking and stores the resulting alignment

SARUMAN: program flow

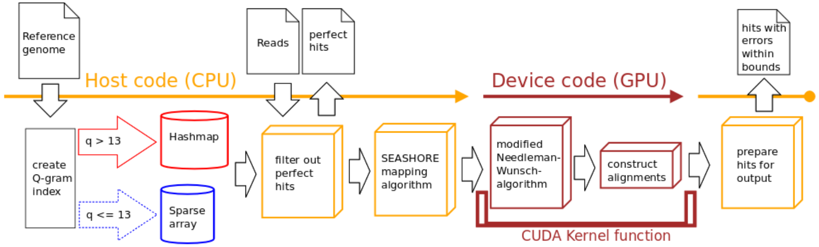


Figure: Overview of SARUMAN's program flow

The q-gram index

Implementation:

- Normal mode: complete genome read at once
- Creating index using hashtable
- For larger genomes: reference genome dividable into several chunks

SEASHORE

- SEASHORE runs on the CPU
- Perfect matches are pre filtered
- Possible hits are collected and saved in memory
- Batch alignment starts if a defined number of hits is reached

Alignment phase

- A large number (e.g. 100.000) of hits is prepared for aligning
- Corresponding genome and read sequences are stored in auxilary data structures
- Memory is allocated on the card for sequences, alignments and scores
- Sequences and parameters are copied to the GPU
- GPU aligns ten-thousands of sequences in parallel

Alignment phase

- During alignment on the GPU the host already collects new hits
- If all alignments are done: data is returned to the host
- Maximal number of hits processable in parallel depends on type of card and VRAM

Results

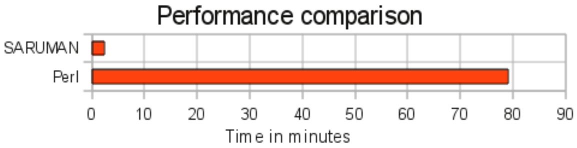
Comparison: Perl implementation $<=>$ SARUMAN

- Bacterial reference genome, 3.65 MB
- Over 6 million Solexa reads
- System: Intel E8400, 3GHz, 8GB RAM, NVidia GTX280
- Perl implementation: 79 minutes
- SARUMAN prototype: 2,3 minutes

Results

Comparison: Perl implementation $<=>$ SARUMAN

- Perl implementation: 79 minutes
- SARUMAN prototype: 2,3 minutes



$\Rightarrow$ over 30$\times$ speedup, reduced memory footprint

Summary

- SEASHORE: exact matching algorithm
- SARUMAN: very fast short read mapping tool
  - All parameters (read length, *e*, alignment cost) variable
  - Complete alignments returned
  - CUDA implementation extremely fast

Thank you for your attention
Questions?