Ultra Fast Sequence Alignment for the DNA Assembly Problem

Michał Kierzynka

Poznań University of Technology

michal.kierzynka@cs.put.poznan.pl

21.03.2013, GTC, San Jose

Outline

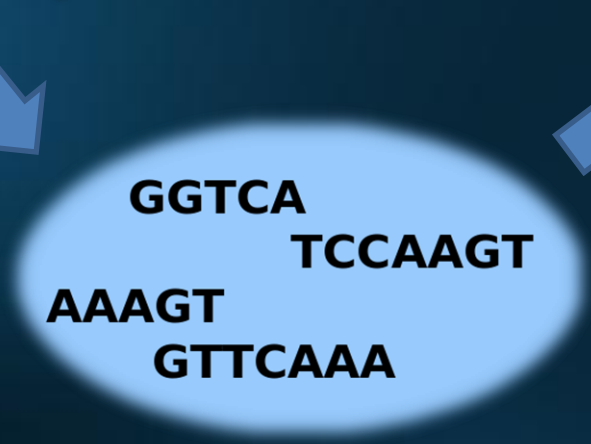
- Introduction to the DNA assembly
- State-of-the-art and motivation
- G-DNA and its optimizations
- Tests results
- Conclusions

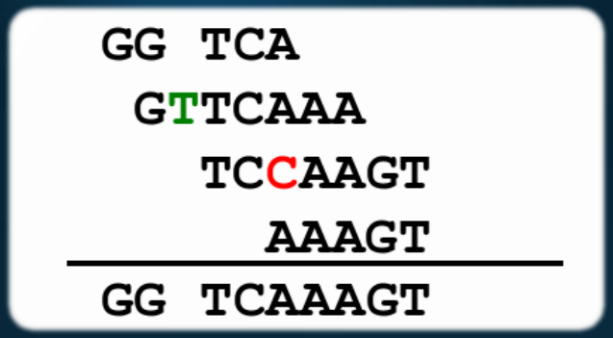
de-novo DNA assembly

DNA *de novo* assembly

- input: short reads (35-150bp)
- output: contigs (assembled parts of a genome)

Illumina Genome Analyzer II sequencer





de-novo DNA assembly

Input sequences:

- a multiset of overlapping reads over alphabet {A, C, G, T}
- may contain misreadings/errors – inexact maches are needed
- come from both strands of DNA helix
- reverse and complementary sequences to consider

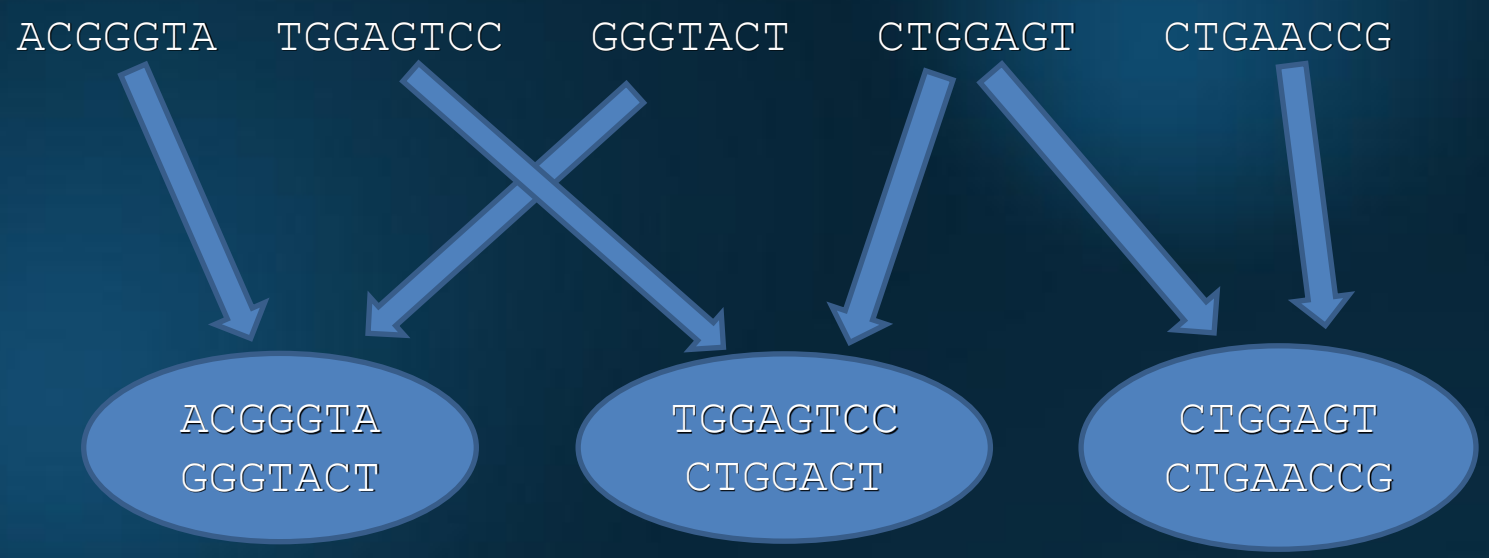
Example reads: AGCA, ATCAAGCAAC, GACTC, TAGAA, TTTGCC

```
TTAGCACAGGAACTCTA
TTTGC-C   GA-CTC
  AGCA       TTCTA
  ATCA-AGCAAC
```

de-novo DNA assembly

The overlap-layout-consensus strategy [1]:

- selection of *promising pairs*



1) Blazewicz, J. and Bryja, M. and Figlerowicz, M. and Gawron, P. and Kasprzak, M. and Kirton, E. and Platt, D. and Przybytek, J. and Swiercz, A. and Szajkowski, L. (2009): Whole genome assembly from 454 sequencing output via modified DNA graph concept. Comput. Biol. Chem., 33(3):224-230

de-novo DNA assembly

The overlap-layout-consensus strategy [1]:

- selection of *promising pairs*
- overlaps verification:
  - sequence alignment (score + shift)

```
ACGGGTA
  GGGTACT        score: 5, overlap 2

CTGGAGT
 TGGAGTCC        score: 6, overlap 1

CTGGAGT
CTGAACCG         score: 1, overlap 0
```

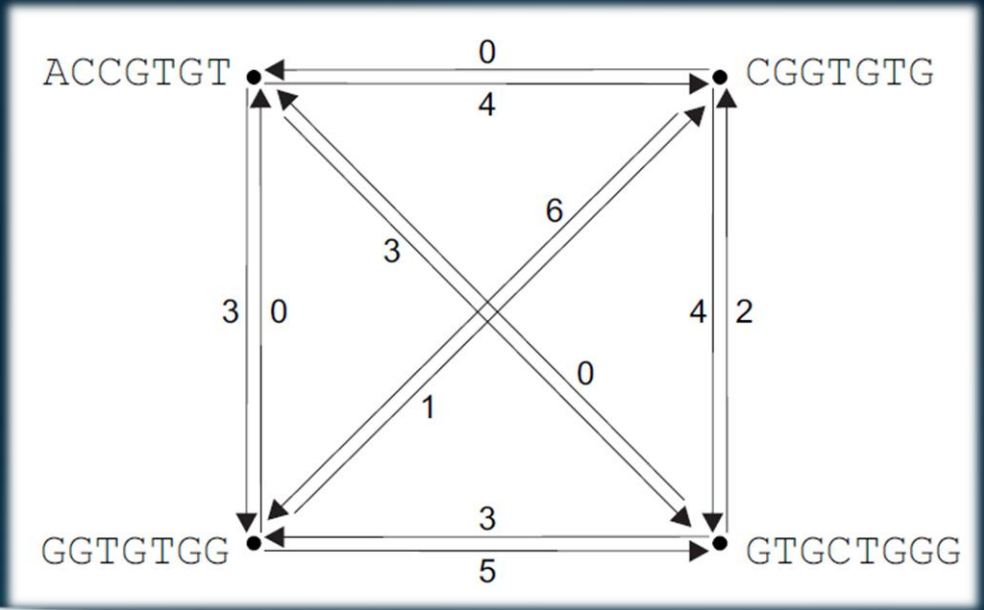
1) Blazewicz, J. and Bryja, M. and Figlerowicz, M. and Gawron, P. and Kasprzak, M. and Kirton, E. and Platt, D. and Przybytek, J. and Swiercz, A. and Szajkowski, L. (2009): Whole genome assembly from 454 sequencing output via modified DNA graph concept. Comput. Biol. Chem., 33(3):224-230

de-novo DNA assembly

The overlap-layout-consensus strategy – the graph model:

- directed weigthed graph
- each read represented by a vertex
- overlapping sequences connected by an arc
- weights – corresponding alignment scores
- result – *minimum path cover* problem (ideally a Hamiltonian path)

Selection of overlapping sequences is the key step!



Motivation

Motivation:

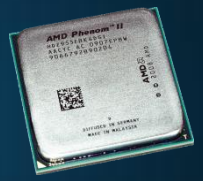
- real-life problem instances are extremely large (e.g. 30M reads)
- sequence alignment takes up to 50% of total time
- exact algorithm (NW) is often replaced by some heuristics

Why to use GPUs?

- they proved to be well suited for sequence alignment

State-of-the-art

A lot of implementations using GPUs, Cell B.E. and SSE instructions.



Drawbacks of the current solutions:

- no support for pairwise alignment of selected pairs
  - most of them support database search only (e.g. CUDASW++2.0, SWIPE)
- usually only SW is implemented
  - results do not include the overlap values (e.g. Farrar, SWIPE)
- usually no optimizations for nucleotide reads

Hence the idea of *G-DNA* (GPU-based DNA aligner)

G-DNA

Assumptions:

- ultra fast alignment of nucleotide reads
- semi-global version of NW
- scoring scheme may be simplified (no need for affine gap penalty)
- output: both scores and shifts

```
TTAGCACAGGAAC-CTA          shift=4
    CACAG-AACTCTAGG        score=9
```

G-DNA = GPU-based DNA Aligner:

- highly optimised for the Fermi architecture
- currently the fastest software in its class worldwide

G-DNA

Sequence data compression:

- each residue uses as few bits as it is required by the cardinality of a given input alphabet

Example:

- 4 residues (A, C, G, T/U)
  - 2 bits per nucleotide =16 symbols per one 32-bit word
- 4 residues + N (uncertain read)
  - 3 bits per nucleotide = 10 symbols per one 32-bit word

Advantages:

- more data may be fetched from the global memory at once
- no need for expensive decompression (simple bitwise operations)

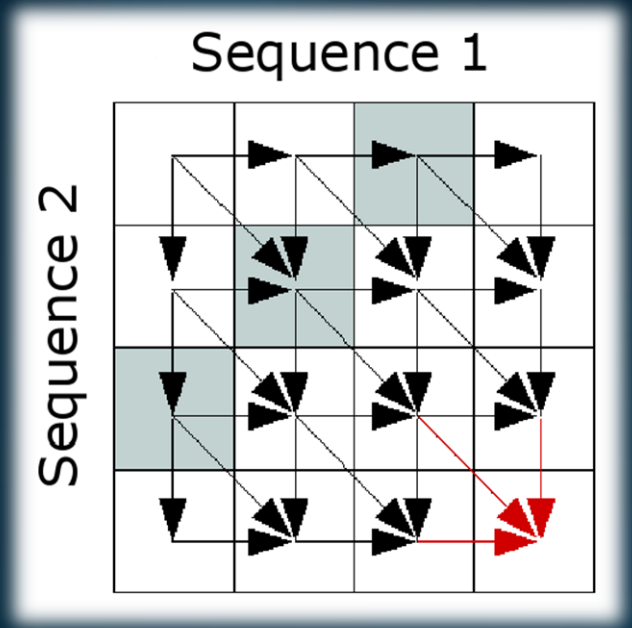
G-DNA

NW and dynamic programming (DP):

- data dependencies: left, upper and diagonal elements are needed

$$H[i,j] = \max \begin{cases} H[i-1,j] - G_{penalty} \\ H[i,j-1] - G_{penalty} \\ H[i-1,j-1] + SM(s_1[i], s_2[j]) \end{cases}$$

Although the diagonal elements may be processed in parallel, this would be highly inefficient wrt. the global memory access.

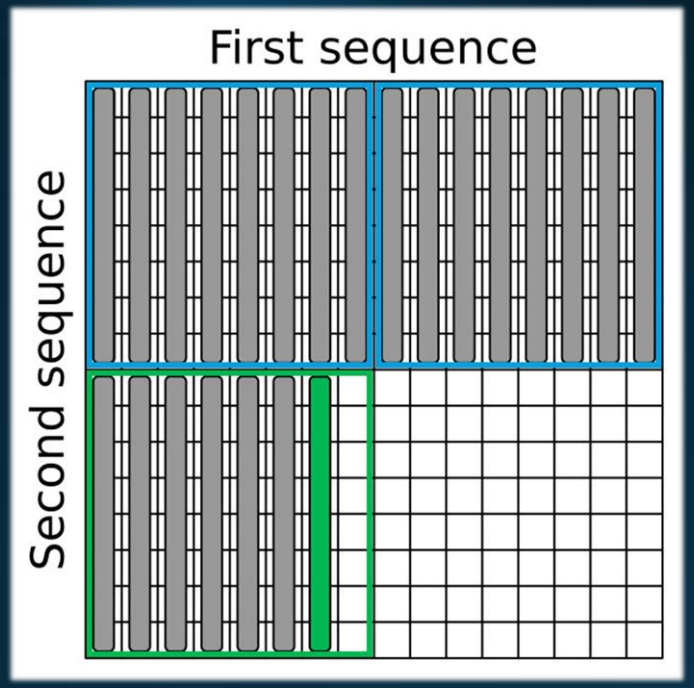


G-DNA

NW and dynamic programming:

- the whole matrix is processed by a single thread
- *MxN* matrix is divided into sub-matricies of *KxK* (*K* is the unroll factor)
  - two most inner loops process a square area of 16x16 (or 10x10) cells
  - cells are processed horizontally in a group of 16 or 10 elements
- up to 256 cells computed from a single fetch

Reduced need for data transfer from the global memory leads to a significant performance boost.



G-DNA

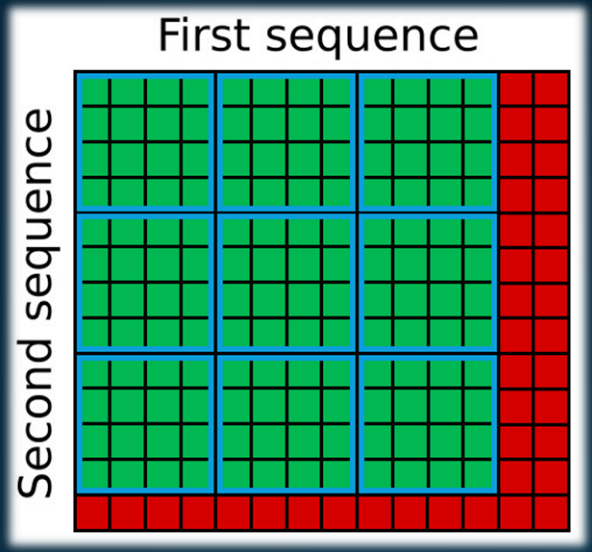
Loop unrolling - crucial for efficiency, especially in case of nested loops (the number of conditional instructions is minimized)

K – the unroll factor is corelated with the number of nucleotides packed within a single 32-bit word, i.e. 16 or 10.

Problem: the code becomes specific to a given sequence length.

Solution: C++ template-based kernels:

- fixed-length reads (16 + 10 kernels)
  - all loops unrolled!
- variable-length reads (2 kernels)
  - only matrix ends not divisible by K are not unrolled



Tests results

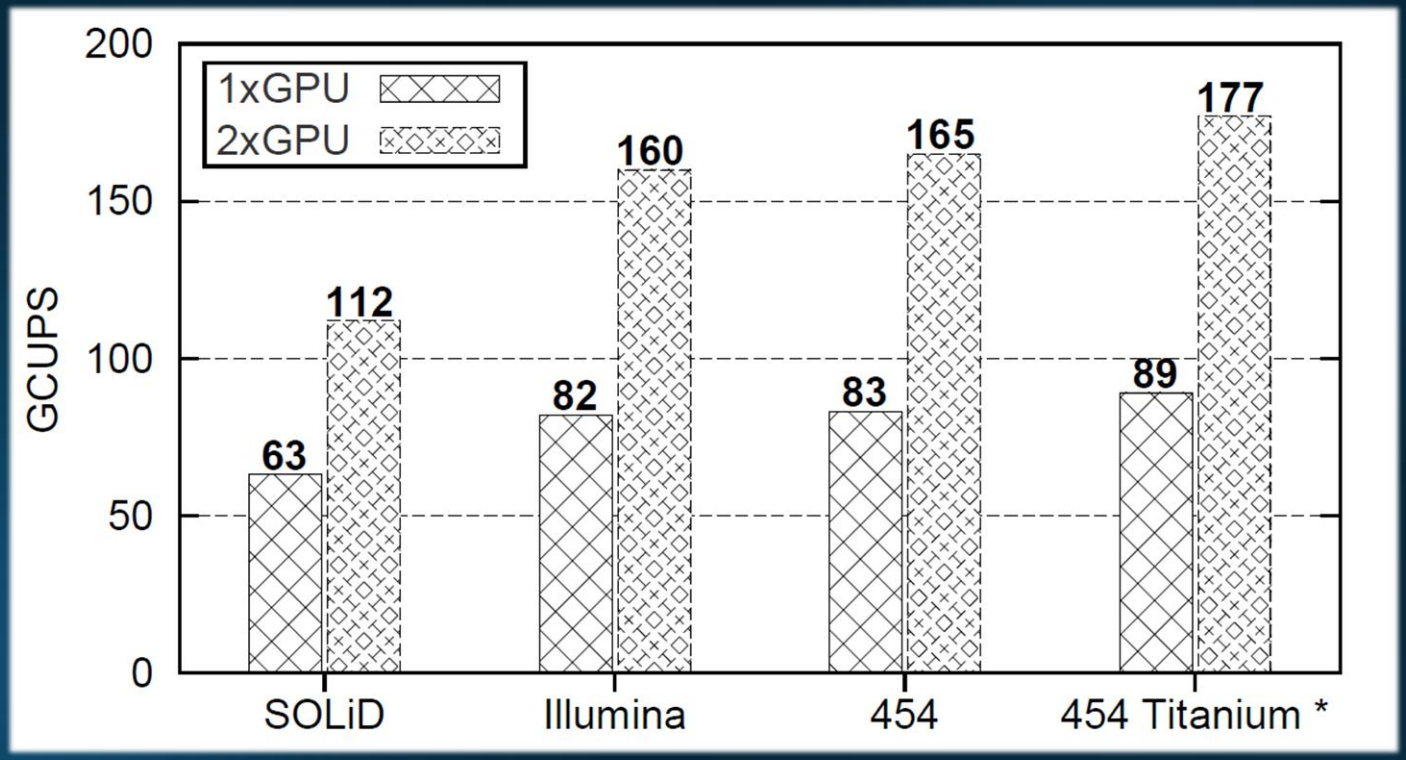
Input data:

- SOLiD: 3.4M reads, 46bp, *Streptococcus suis*
- Illumina GA IIx: 34M reads, 120bp, *Clonorchis sinensis*
- Roche 454: 436k reads, avg. 235bp, *E. coli*
- Roche 454 GS FLX Titanium: 1020bp, to test peak performance

Hardware:

- GPU: 2 x NVIDIA GeForce GTX 580
- CPU: Intel Core 2 Quad Q8200, 2.33GHz
- RAM: 8GB

Tests results



GCUPS – Giga Cell Updates Per Second

* refers to long reads only

Tests results

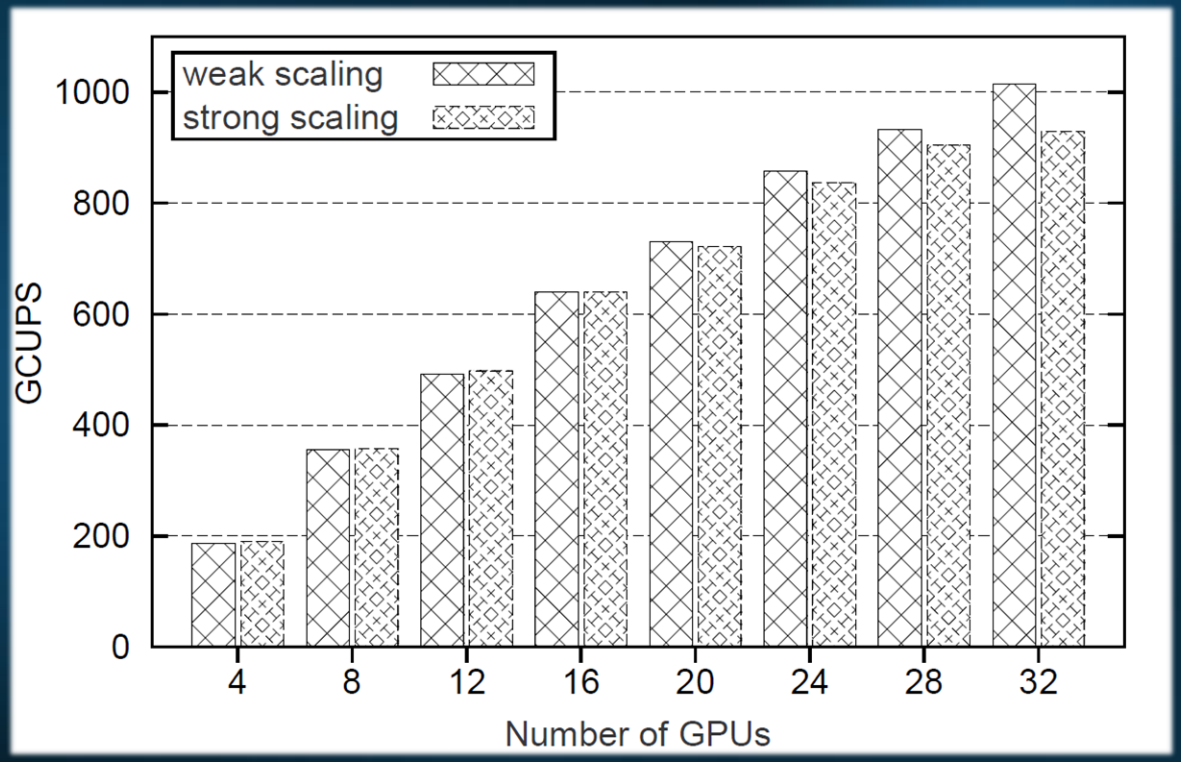
89 GCUPS on a single GPU makes G-DNA quite fast:

- GPU
  - CUDASW++2.0: up to 48 GCUPS on GeForce GTX 580
  - Ligowski & Rudnicki's approach: 43 GCUPS on GeForce GTX 480
  - 160 GCUPS on Tesla K10 with Vector Video Instructions
- CPU
  - Farrar's STRIPED: 20 GCUPS, 8 cores
  - SWIPE: 53 GCUPS on Intel Xeon X5650, 6 cores
- Cell B.E.
  - Farrar's STRIPED: 15.5 GCUPS on IBM QS20
  - SWPS3: 8 GCUPS on PS3

Tests results

MPI version of G-DNA:

- the weak scaling test: 1014 GCUPS for 110M seqs. (32 GPUs)
- the strong scaling test: 929 GCUPS (problem size fixed at 55M seqs.)



32 nodes, each with a single Tesla M2050

a real-life use case

A real-life example:

- 20M paired-end reads coming from the Illumina GAII sequencer
- 40M reads in total (including reverse complementary reads)

G-DNA used to find promising (similar) sequences:

- needs 157 minutes to find ~300M pairs of highly similar sequences
  - using ~100 GCUPS of average performance
- comparing every sequence witch each other would take decades, even on a HPC cluster
- heuristics pointing pairs of sequences to verify are out of the scope of the presentation

Conclusions

- G-DNA – a highly efficient tool for aligning nucleotide reads
- designed for the DNA assembly problem
- performance:
  - ultra fast implementation of NW
  - support for multiple GPUs
  - immensely quick on computational clusters >1 TCUPS
- an ongoing work: application of G-DNA in an algorithm for DNA *de novo* assembly