Local sequence alignment by using on-chip parallelism

Student: Artem Kupchinskiy, SpbAU

Instructor: Alexander Tiskin, Warwick University, UK

Path to vectorization

Classic pipeline: data -> algorithm -> result

Poor perfomance is commonly treated by some code optimization:

- pass by reference instead of value
- use the quick sort instead of bubble one
- cache intermediate computations

However source code does not provide the full story!

Vectorization: intro

Vecorization is a thing lying beyond the source code

Vector processor - a processor supporting operations with one-dimensional arrays

Such processors could improve algorithm on the PC architecture level. They provide instruments for simultaneous processing several scalar operations.

Scalars versus vectors

```
void addindex(float *x, int n) {
  for (int i = 0; i < n; i++)
    x[i] = x[i] + i;
}
```

the source of efficiency

```
#include <ia32intrin.h>

// n a multiple of 4, x is 16-byte aligned
void addindex_vec(float *x, int n) {
  __m128 index, x_vec;

  for (int i = 0; i < n/4; i++) {
    x_vec = _mm_load_ps(x+i*4);                    // load 4 floats
    index = _mm_set_ps(i*4+3, i*4+2, i*4+1, i*4); // create vector with indexes
    x_vec = _mm_add_ps(x_vec, index);              // add the two
    _mm_store_ps(x+i*4, x_vec);                    // store back
  }
}
```

In general, vectorization reduces a number of iterations (the inner cycle complexity could get higher)

The LCS Problem

- Two strings A and B
- The longest common sequence (LCS) score: the length of the longest string that is subsequence of both A and B

Example:
A = ACCCCA, B = CGGGCGGGCGGGCGG

LCS(A, B) = 4

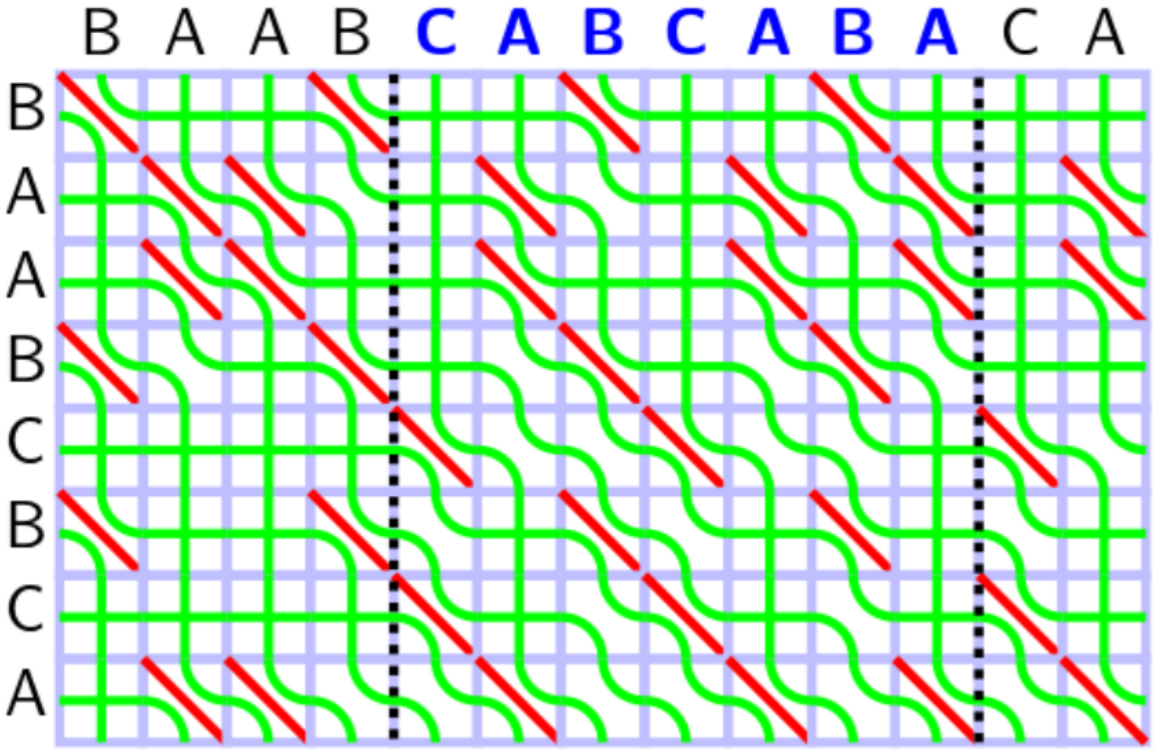
The semi-local LCS problem

- Give the (implicit) matrix of LCS scores
- String-substring LCS: string a vs every substring of b
- Prefix-suffix LCS: every prefix of a vs every suffix of b
- Suffix-prefix LCS: every suffix of a vs every prefix of b
- Substring-string LCS: every substring of b vs string a

Q1: Can we solve this more general problem?
Q2: Can we vectorize our algorithm?

The seaweed method

Using the features of highly symmetrical objects (related to Braid group) it is possible to build the quadratic algorithm, which can be vectorized. The visualization of the structure is below:



What's this have to do with biology?

2010: E. Picot, P. Krusche, A. Tiskin, I. Carré, and S. Ott. Evolutionary Analysis of Regulatory Sequences (EARS) in Plants.

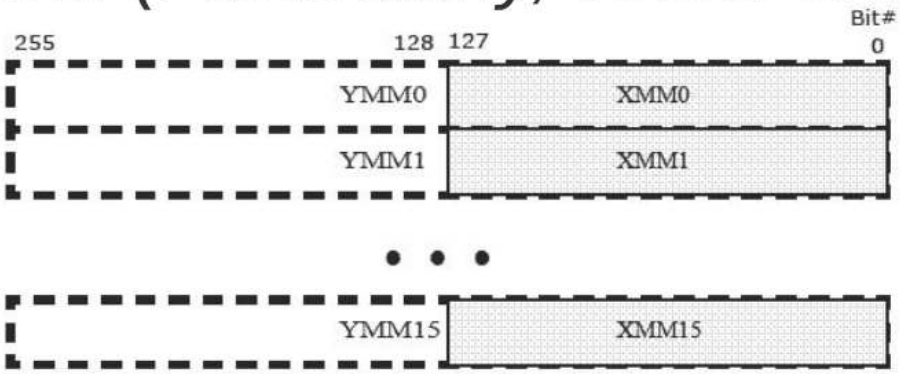
2012: L. Baxter, A. Jironkin, R. Hickman, J. Moore, C. Barrington, P. Krusche, N. P. Dyer, V. Buchanan-Wollaston, A. Tiskin, J. Beynon, K. Denby, and S. Ott. Conserved Noncoding Sequences Highlight Shared Components of Regulatory Networks in Dicotyledonous Plants.

The algorithm was applied for detection of evolutionarily conserved sequences in the first work.
Also it helps to find similarity between orthologous promoters in the second research.

The goal of the project

Investigate the potential of modern processors (supporting AVX instruction sets)
to speed up the alignment algorithm.

Challenges: AVX doesn`t support real physical 256-bit registers. There are two 128-bit ones indeed. Thus some basic operations cost more than it expected. (Particularly, vector shifts)



The final results

Good news:

- Some parallalelized version of the seaweed algorithm was coded and showed speed-up about 1.7 in comparison with the scalar version
- Useful experience of mid-level coding.

Bad news:

- Potential of AVX-512 was not tested
- My realization of parallelized version is far from optimal (but improvements require some efforts)
- 1.7 is not a big factor of speed-up

The end