

White paper on CLC Bioinformatics Cube version 0.2.0

1. Introduction

CLC Bioinformatics Cube is a small hardware device used for high performance database searches – searches that are designed to perform at very high speeds without compromising the quality of the search results. This is contrary to the very popular BLAST searches, where focus is on speed at the expense of search result quality.

The Cube is a 12 cm x 12 cm x 12 cm steel cube containing two Field Programmable Gate Array (FPGA) Chips mounted on boards with external memory.

The boards are hooked up through a USB connection to the computer.



The Cube is designed with the following in mind:

- It should be fast
- It should be easy to use
- It should be easy to integrate in an existing IT infrastructure

FPGA chips have the advantages of being reconfigurable and able to perform calculations in a highly parallelized fashion, resulting in very fast calculations, compared to normal computers. However, it is challenging to program the chips effectively, so it takes a lot of effort to implement good algorithms.

To make the CLC Bioinformatics Cube easy to use, it is hooked up to the computer by use of a USB connection. An important aspect of user friendliness is, of course, the software. Here, we have chosen to provide two ways of accessing the Cube. It can be done either via the command line, or through our graphical user interface of CLC bio's other bioinformatics software applications, CLC Gene Workbench, CLC Protein Workbench, and CLC Combined Workbench. The command line version is based on NCBI BLAST, so it has almost the same options, input format, and output formats.

---- 000 ----



We have created a Cube which runs Smith Waterman up to 125 times faster than a 3 GHz desktop computer, and acceptably close to the speed of BLAST. Smith Waterman searches taking 14 hours on the desktop are thus reduced to around 7 minutes on the Cube.

This is a breakthrough for desktop-based bioinformatics, as it gives researchers the possibility of running BLAST-like searches of a significantly higher quality than before.

Our comparison between BLAST and Smith Waterman thus shows that running BLASTp with the default expectation value of 10 often results in only half as many hits as running a Smith Waterman search on the Cube.

The risk of missing important information when BLASTing is thus very high. In our opinion, a strategy of BLASTing proteins instead of doing Smith Waterman searches on small to medium-sized datasets should therefore be considered carefully before being executed.

On larger searches¹ the amount of missing hits on BLASTp is around 12 percent, resulting in almost 900 additional hits when searching the dataset by means of the Cube, compared to searching with the standard NCBI BLASTp.

BLASTing with lower e-values reduces the number of missing hits compared to using the Cube. This does, however, not solve the problem of poor quality data from BLAST searches, as the total number of hits is also reduced significantly when BLASTing with lower e-values.

1.1 Requirements

To get the best result from the CLC Cube, you should have the following:

- A PC with Windows XP or Linux, or an Apple computer with Mac OS X
- A 1.5 GHz or faster Pentium CPU (or PowerPC Mac G4)
- A version 2.0 USB connection

This White Paper benchmarks the speed of the Cube against the speed of a DELL Desktop with a 3.0 GHz Pentium 4 CPU, and 2 GB of RAM.

2. About the algorithms

CLC Cube version 0.2.0 enables accelerated database searching using the Smith Waterman algorithm.

Basically, the NCBI BLAST (ver. 2.2.14) has been rewritten so that the database search parts of the BLAST algorithm have been replaced with the Smith Waterman algorithm.

The result is a database search with an interface like BLAST, but at the same time a database search that - contrary to BLAST searches - guarantees to find the optimal local alignment between the sequences.

¹ Searches where the total number of pairwise amino acid comparisons (query bases x database bases) is around 20,000,000,000 bases

The main differences between BLAST and the Smith Waterman algorithm are these:

- The Smith Waterman algorithm guarantees to find the optimal local alignment between the sequences, whereas BLAST only approximates this.
- The Smith Waterman algorithm only returns a single best alignment for each pair of sequences being compared, whereas the BLAST algorithm may return several alignments.
- To guarantee the optimal alignments, the Smith Waterman algorithm needs to perform many more computations than the BLAST algorithm, making it significantly slower.

2.1 When to use BLAST

BLAST is very useful if you are looking for close matches (contrary to best matches) and/or you do not mind missing lower homology sequences.

BLAST is also very useful if you want a quick answer.

2.2 When to use Smith Waterman

Contrary to BLAST, Smith Waterman guarantees that you will find the optimal local alignment, and Smith Waterman thus provides significantly more precise and therefore better results than BLAST.

Smith Waterman is, however, much slower than BLAST; it is therefore only the best algorithm to use if fully correct answers are more important than quick answers.

2.3 Summary

On an overall level, you can say that

- BLAST is the better in the explorative phases of the research, and Smith Waterman is the better in the late stages of the research, where absolutely correct results are more important than estimates.
- If you have query sequence data with too many of inherent errors, BLAST searches would often be so imprecise that the results are wrong or misleading.

The Cube significantly changes this picture by allowing **database** searches at a high quality level within reasonable execution times.

3. The workflow

This is the workflow in a Cube Smith Waterman BLAST search:

1. The user defines the query sequence dataset, the database to be searched, the e-value, and the scoring matrix to be used.
2. The Smith Waterman BLAST is started through the command line interface (exactly like NCBI's BLAST command line interface) or through the graphical user interface of CLC bio's bioinformatics software applications. If one of these is used, the results can be shown either graphically in a table or as text corresponding to the traditional BLAST output.
3. The final output is a number of pairwise alignments between the query sequences and the database sequences. Among others, the number of aligned sequences depends on the e-value.

The Cube performance is best at low e-values. The reason for this is that the alignment part of the Smith Waterman BLAST is made in the software, which by nature is not sped up in the Cube.

3.1 Types of database searches

Types of Smith Waterman based BLAST searches which can be run on the Cube:

| Type of search | Query type | Database type | Comparison | Note |
|----------------|------------|---------------|-----------------------|--|
| BLASTn | Nucleotide | Nucleotide | Nucleotide-Nucleotide | |
| BLASTp | Protein | Protein | Protein-Protein | |
| tBLASTn | Protein | Nucleotide | Protein-Protein | The database is translated into protein |
| BLASTx | Nucleotide | Protein | Protein-Protein | The queries are translated into protein |
| tBLASTx | Nucleotide | Nucleotide | Protein-Protein | The queries and the database are translated into protein |

4. Introduction to the benchmarks

Below, we have compared the following platforms

- CLC Bioinformatics Cube
- clcbast_0.2.0
- A DELL Desktop with a 3.0 GHz Pentium 4 CPU, and 2 GB of RAM

Smith Waterman BLASTn was tested on a dataset consisting of all horse EST sequences in GenBank (36,914 sequences with a total number of residues of 19,762,562).

Smith Waterman BLASTp was tested on a dataset consisting of 50,000 randomly chosen protein sequences from SwissProt (the total number of residues is 18,396,764).

In the following, the concept “speed-up” is used for describing the execution time of the algorithm without using the Cube, compared to the execution time of the algorithm when the Cube is plugged in.

4.1 Benchmark of Smith Waterman BLASTn

From table 3 below it can be seen that the Cube speed is 4-8 times faster than the 3 GHz Desktop at small amounts of query sequences and small datasets.

On larger amounts of query sequences, and especially on larger datasets, the speed-up increases. With databases above 500,000 nucleotides, the Cube speed is between 35 and 128 times faster than a 3 GHz Desktop. At e-values at 1 or below, the speed-up is always above 78 times.

The Cube runs at a speed of up to 5 Giga Cell Updates Per Second (GCUPS), compared to speeds of up to 0.05 GCUPS on the 3GHz Desktop.

The number of query sequences has some significance, but not much.

The reason for the low speed-up at high e-values and with small databases is the time overhead incurred by the parts of the searches performed in the software. This will be improved significantly in future versions of the software.

4.2 Benchmark of Smith Waterman BLASTp

Because there are only 4 different nucleotides but 20 different amino acids, the speed of the Cube is reduced when doing amino acid searches to nucleotide searches. Due to the construction of desktop computers and laptop computers, their speed is, however, not reduced significantly when performing amino acid searches.

Table 4 below shows that the Cube speed is around 3 to 9 times faster than the 3 GHz Desktop in connection with small amounts of query sequences and small datasets.

In connection with larger amounts of query sequences, and especially with larger datasets, the speed-up increases. At databases above 350,000 amino acids, the Cube speed is from 25 to 35 times faster than a 3 GHz Desktop.

The Cube runs at a speed of up to 1.55 GCUPS, compared to speeds of up to 0.046 GCUPS on the 3GHz Desktop.

The number of query sequences has some effect, but not much.

Table 1. Test data – Benchmark of Smith Waterman BLASTn

| Name in table below | Number of sequences | Total number of residues | Note |
|---------------------|---------------------|--------------------------|--|
| All | 36,914 | 19,762,562 | All horse EST sequences in GenBank |
| 1000 | 1,000 | 533,828 | 1,000 randomly chosen horse EST sequences in GenBank |
| 100 | 100 | 52,489 | 100 randomly chosen horse EST sequences from the 1,000 above |

Table 2. Test data - Benchmark of Smith Waterman BLASTp

| Name in table below | Number of Sequences | Total number of Residues | Note |
|---------------------|---------------------|--------------------------|---|
| 50000 | 50,000 | 18,396,764 | 50,000 randomly chosen sequences from SwissProt |
| 1000 | 1,000 | 357,564 | 1,000 randomly chosen sequences from the 50,000 above |
| 100 | 100 | 33,701 | 100 randomly chosen sequences from the 1,000 above |

Table 3. Benchmarks – Smith Waterman BLASTn

| No. of query sequences | No of seq. in database | E-value | Smith Waterman on 3 GHz desktop | | | | Quality (number of hits) | | | Time for NCBI BLASTn on 3 GHz desktop | |
|------------------------|------------------------|---------|---------------------------------|--------------|-----------------|-----------------|--------------------------|--------|--------|---------------------------------------|------------------------|
| | | | On Cube | GCUPS (Cube) | No Cube | GCUPS (no Cube) | Speed-up | Cube | BLASTn | | Quality index Cube=100 |
| 100 | 100 | 10.00 | 0m 24.9s | 0,22 | 1m 54.0s | 0,05 | 4.6 x | 2.073 | 1.840 | 88.8 | 0.8s |
| 100 | 1000 | 10.00 | 0m 34.9s | 1,61 | 20m 22.3s | 0,05 | 35.0 x | 2.639 | 2.444 | 92.6 | 1.1s |
| 100 | All | 10.00 | 7m 31.2s | 4,60 | 13h 49m 27.8s | 0,04 | 110.3 x | 16.044 | 15393 | 95.9 | 14.7s |
| 1000 | 100 | 10.00 | 3m 57.5s | 0,24 | 21m 45.0s | 0,04 | 5.5 x | 19.459 | 17.070 | 87.7 | 6.7s |
| 1000 | 1000 | 10.00 | 7m 13.9s | 1,31 | 4h 23m 1.0s | 0,04 | 36.4 x | 26.433 | 24.877 | 94.1 | 11.4s |
| 1000 | All | 10.00 | 79m 20.8s | 4,43 | Still running ☺ | | | | | | 2m 30.3s |
| 1000 | 1000 | 0.01 | 2m 34.2s | 3,70 | 4h 6m 40.7s | 0,04 | 96.1 x | 3.969 | 3.963 | 99.8 | 5.1s |
| 1000 | 1000 | 0.10 | 2m 41.6s | 3,53 | 4h 14m 13.4s | 0,04 | 94.4 x | 4.873 | 4.780 | 98.1 | 5.3s |
| 1000 | 1000 | 1.00 | 3m 14.9s | 2,92 | 4h 25m 19.6s | 0,04 | 81.7 x | 8.706 | 8.357 | 96.0 | 6.4s |
| 1000 | 1000 | 10.00 | 7m 13.9s | 1,31 | 4h 23m 1.0s | 0,04 | 36.4 x | 26.433 | 24.877 | 94.1 | 11.4s |
| 100 | 1000 | 0.01 | 0m 14.6s | 3,84 | 21m 43.6s | 0,04 | 89.0 x | 369 | 369 | 100.0 | 0.6s |
| 100 | 1000 | 0.10 | 0m 15.3s | 3,66 | 22m 10.2s | 0,04 | 87.0 x | 464 | 455 | 98.1 | 0.5s |
| 100 | 1000 | 1.00 | 0m 17.0s | 3,30 | 22m 15.7s | 0,04 | 78.4 x | 892 | 837 | 93.8 | 0.6s |
| 100 | 1000 | 10.00 | 0m 34.9s | 1,61 | 20m 22.3s | 0,05 | 35.0 x | 2.639 | 2.444 | 92.6 | 1.1s |
| 100 | 100 | 0.01 | 0m 3.9s | 1,42 | 1m 57.9s | 0,05 | 30.3 x | 122 | 122 | 100.0 | 0.2s |
| 100 | 100 | 0.10 | 0m 4.1s | 1,34 | 1m 57.7s | 0,05 | 28.6 x | 162 | 158 | 97.5 | 0.2s |
| 100 | 100 | 1.00 | 0m 6.2s | 0,32 | 2m 2.0s | 0,05 | 19.6 x | 346 | 321 | 92.8 | 0.3s |
| 100 | 100 | 10.00 | 0m 24.9s | 0,16 | 1m 54.0s | 0,05 | 4.6 x | 2.073 | 1.840 | 88.8 | 0.8s |
| 100 | All | 0.01 | 6m 50.8s | 5,05 | 14h 26m 12.8s | 0,04 | 126.5 x | 7.254 | 7244 | 99.9 | 13.0s |
| 100 | All | 0.10 | 6m 52.1s | 5,03 | 14h 36m 17.0s | 0,04 | 127.6 x | 7.836 | 7816 | 99.7 | 11.3s |
| 100 | All | 1.00 | 6m 58.5s | 4,96 | 14h 37m 22.9s | 0,04 | 125.8 x | 9.720 | 9580 | 98.6 | 12.1s |
| 100 | All | 10.00 | 7m 31.2s | 4,60 | 13h 49m 27.8s | 0,04 | 110.3 x | 16.044 | 15393 | 95.9 | 14.7s |

All times are exclusive of printing to output files or to the screen.

Table 4. Benchmarks - Smith Waterman BLASTp

| No. of query sequences | No of seq. in database | E-value | 3 GHz desktop | | | | Quality (number of hits) | | | Time for NCBI BLASTp on 3 GHz desktop | |
|------------------------|------------------------|---------|---------------|--------------|--------------|-----------------|--------------------------|--------|--------|---------------------------------------|------------------------|
| | | | Cube | GCUPS (Cube) | No Cube | GCUPS (no Cube) | Speed-up | Cube | BLASTp | | Quality Index Cube=100 |
| 100 | 100 | 10.00 | 12.0s | 0,09 | 34.2s | 0,03 | 2.9 x | 1.040 | 597 | 57.4 | 1.3s |
| 100 | 1000 | 10.00 | 32.2s | 0,37 | 4m 45.4s | 0,04 | 8.9 x | 1.577 | 810 | 51.4 | 2.4s |
| 100 | 50000 | 10.00 | 7m 18.1s | 1,42 | 4h 4m 7.5s | 0,04 | 33.9 x | 6.961 | 6.075 | 87.3 | 1m 8.7s |
| 1000 | 1000 | 10.00 | 5m 35.4s | 0,38 | 1h 1m 50.9s | 0,03 | 11.1 x | 16.466 | 8.203 | 49.8 | 24.9s |
| 1000 | 1000 | 0.01 | 1m 39.0s | 1,29 | 44m 41.6s | 0,05 | 27.1 x | 2.142 | 2.091 | 97.6 | 18.9s |
| 1000 | 1000 | 0.10 | 1m 43.1s | 1,24 | 45m 44.5s | 0,05 | 26.6 x | 2.481 | 2.314 | 93.3 | 19.1s |
| 1000 | 1000 | 1.00 | 2m 13.1s | 0,96 | 53m 3.1s | 0,04 | 23.9 x | 4.056 | 3.149 | 77.6 | 20.0s |
| 1000 | 1000 | 10.00 | 5m 35.4s | 0,38 | 1h 1m 50.9s | 0,03 | 11.1 x | 16.466 | 8.203 | 49.8 | 24.9s |
| 100 | 100 | 0.01 | 2.7s | 0,42 | 25.1s | 0,05 | 9.3 x | 106 | 103 | 97.2 | 0.5s |
| 100 | 100 | 0.10 | 3.0s | 0,38 | 25.9s | 0,04 | 8.6 x | 120 | 113 | 94.2 | 0.6s |
| 100 | 100 | 1.00 | 4.8s | 0,24 | 25.6s | 0,04 | 5.3 x | 223 | 168 | 75.3 | 0.6s |
| 100 | 100 | 10.00 | 12.0s | 0,09 | 34.2s | 0,03 | 2.9 x | 1.040 | 597 | 57.4 | 1.3s |
| 100 | 1000 | 0.01 | 9.5s | 1,27 | 4m 7.5s | 0,05 | 26.0 x | 199 | 195 | 98.0 | 1.9s |
| 100 | 1000 | 0.10 | 10.1s | 1,19 | 4m 9.7s | 0,05 | 24.7 x | 231 | 216 | 93.5 | 1.8s |
| 100 | 1000 | 1.00 | 12.7s | 0,95 | 4m 16.4s | 0,05 | 20.2 x | 369 | 294 | 79.7 | 1.9s |
| 100 | 1000 | 10.00 | 32.2s | 0,37 | 4m 45.4s | 0,04 | 8.9 x | 1.577 | 810 | 51.4 | 2.4s |
| 100 | 50000 | 0.01 | 6m 38.5s | 1,56 | 3h 44m 39.9s | 0,05 | 33.9 x | 4.049 | 3.993 | 98.6 | 1m 8.2s |
| 100 | 50000 | 0.10 | 6m 40.4s | 1,55 | 3h 46m 57.1s | 0,05 | 34.0 x | 4.379 | 4.315 | 98.5 | 1m 6.1s |
| 100 | 50000 | 1.00 | 6m 47.4s | 1,52 | 3h 52m 23.1s | 0,04 | 34.3 x | 4.952 | 4.796 | 96.8 | 1m 6.7s |
| 100 | 50000 | 10.00 | 7m 18.1s | 1,42 | 4h 4m 7.5s | 0,04 | 33.9 x | 6.961 | 6.075 | 87.3 | 1m 8.7s |

All times are exclusive of printing to output files or to the screen.

4.3 tBLASTn, BLASTx, and tBLASTx

The speed-up of the Smith Waterman versions of tBLASTn, BLASTx, and tBLASTx is approximately the same as for the Smith Waterman version of BLASTp.

The reason is that these algorithms also perform protein-protein comparisons.

5. The basics of FPGA technology

When designing traditional integrated circuits, one connects various gates (OR gates, AND gates, inverters, etc.) in a large grid that ends up performing the desired function. This is how CPUs in computers are made. Generally, computer CPUs are designed to be able to do many different things, such as adding integer numbers, multiplying real numbers, reading and writing to memory, etc. The FPGA technology used in the CLC Cube is also based on a large number of gates in an integrated circuit, but rather than making general calculations like CPUs, its function is to simulate how an integrated circuit works.

This means that you can design an integrated circuit and upload it to the FPGA. The FPGA then simulates how the circuit works, basically making the FPGA work like a specially designed chip. The unique advantage to this approach is that any number of circuit designs can be uploaded to the FPGA, giving it different functions.

Programming the FPGA is actually not done with normal gates, which typically have two input wires. Instead, the FPGA uses lookup tables with four inputs, making the individual units a little more advanced than normal gates. The output from these lookup tables can then be connected to the input wires for other tables in a large network, just like standard integrated circuits. FPGA chips also have a small amount of internal memory which can be used for remembering various states and for storing some data. By programming the FPGAs in this way, it is possible to give it almost any function.

The specific circuit design uploaded to the FPGA determines at which clock speed the simulation can be run. If the circuit has many tables and connections that need to be traversed in each calculation, the clock speed will be lower than if the circuit is simpler. Thus, an FPGA does not have a fixed clock speed, but rather a programmable one, which is adjusted to suit the specific calculations to be performed.

5.1 FPGA technology and the CLC Cube

The CLC Cube contains two FPGA chips mounted on boards with external memory. The internal memory of a chip is very limited, so adding the extra external memory to the boards allows for larger amounts of data to be stored. The boards are connected through a USB connection to the computer. The current version 2.0 of USB connections has quite a high capacity, but given the very high computing power of the FPGAs, it is always a challenge to design the algorithms to fit the USB bandwidth.

5.2 Smith Waterman

The Smith Waterman implementation in the CLC Cube is based on a circuit design with many nodes, each of which corresponds to an entry in the dynamical programming matrix used in the Smith Waterman algorithm. For each clock cycle of the FPGA, each of the nodes is updated, based on the values in the neighboring matrix entries. To avoid unnecessary dependencies between the matrix entries being updated in a given clock cycle, the nodes correspond to entries in a diagonal of the dynamical programming matrix. Each clock cycle of the FPGA then updates a new diagonal in the matrix.

The clock speed of an FPGA is typically quite low compared to CPUs, e.g. in the order of 50 MHz. There are two reasons why FPGAs can still attain a very high performance. One is that complex calculations can be done in a single clock cycle, e.g. filling out a Smith Waterman matrix entry. The other reason for the high performance is the possibility to parallelize the calculations, e.g. filling out multiple Smith Waterman matrix entries at a time.

A typical speed calculation looks like this:

3.0 GHz Pentium 4 CPU: $3.0 \text{ Ghz} / 30 \text{ cycles per entry} = 100 \text{ MCUPS}$ (mega cell updates per second)

FPGA: $50 \text{ MHz} \times 50 \text{ entries per cycle} = 2500 \text{ MCUPS}$

6. References

Temple F. Smith and Michael S. Waterman, "Identification of Common Molecular Subsequences", *Journal of Molecular Biology*, 147:195-197, 1981.