SmithWaterman-CUDA 1.92 User's Guide

Svetlin Manavski [svetlin.a@manavski.com]

September 14, 2008

1 Introduction

In the Bioinformatics industry searching similarities in protein and genomic databases has become a routine procedure while the amount of data being sequenced and made accessible for analysis is doubling every 12 months.

The Smith-Waterman algorithm [1], available for more than 25 years, is the only one guaranteed to find the optimal local alignment.

SmithWaterman-CUDA allows to perform alignments between one or more sequences and a database (all the sequences, even in the DB, are intended to be proteinic).

The application dynamically performs load balancing among all the computational devices available on the machine. The user can decide how and which resources to use as explained in 4.2.

SmithWaterman-CUDA computes the maximum value of the alignment between each query sequence and each sequence in the given database.

2 System requirements

2.1 Hardware

SmithWaterman-CUDA has some harware requirements that need to be ensured before installing the application. This is the minimal configuration needed:

SmithWaterman-CUDA hardware requirements	
GPU	NVIDIA GeForce 8600 or superior
CPU	Dual core
RAM	1 GB
MOTHER BOARD	Pci-Express
HARD DISK	200 MB of free space

2.2 Software

SmithWaterman-CUDA has also some software requirements that need to be ensured before installing the application.

SmithWaterman-CUDA **Software Requirements**	
OPERATING SYSTEM	Unix & Windows
DISTRIBUTION	- Linux Fedora Core 7 - Linux Fedora Core 8 - OpenSUSE 10.1, 10.2, 10.3 - Ubuntu 7.04, 7.10 - Red Hat Enterprise Linux 3 - Red Hat Enterprise Linux 4 - Red Hat Enterprise Linux 5.0 (32-bit and 64-bit) - SUSE Linux Enterprise Desktop 10.0 - Windows XP
LIBRARIES	Qt4
OTHER	NVIDIA Driver with CUDA Support (169.09) (display driver) CUDA Toolkit version 1.1

For the installation of the CUDA run-time see A.

3 Installation

The installation involves 3 elements: the executable swcuda, a folder *lib* with 3 files (*libQtCore.so.4, libQtGui.so.4, libQtGui.so.4.2.2*) and a configuration file named *config.ini.*

Put all of them into a new directory named swcuda. Then set and export the *LD_LIBRARY_PATH* variable in the following way:

```
LD_LIBRARY_PATH=/home/username/swcuda/lib:/usr/local/cuda/lib/
export LD_LIBRARY_PATH
```

Above we have supposed two things:

1. the swcuda directory is placed in the user personal folder
2. CUDA has been installed in /usr/local

If in your case there is something different, change the *LD_LIBRARY_PATH* setting in the right way.

4 Getting started

A complete comprehension of SmithWaterman-CUDA functionalities can be obtain through an exhaustive explanation of the command line options and of the configuration file that controls the application.

4.1 Command Line Options

SmithWaterman-CUDA has three simple command line options. The application can be run with the following command:

./swcuda query_sequences_file database_file offset

Here there is an explanation:

1. *query_sequences_file*: it is the file containing the sequences (at least one) to align. It must be in fasta format and in the same directory of the executable. If this option is not provided the application enters the interactive mode and asks directly to the user to type the name of the file.

2. *database_file*: it is the database containing the sequence against which those in the *query_sequences_file* have to be aligned. It must be in fasta format and in the same directory of the executable. If this option is not provided the application enters the interactive mode and asks directly to the user to type the name of the file.

3. *offset*: the user can decide to start the alignment not from the first sequence in the *query_sequences_file* but from the offset one. The sequences numeration is intended to start from zero.

All the output scores are saved in a single file but divided according to the query sequence. The output file, that can be found into the output directory (4.2), has a name composed by the *query_sequences_file* plus the *database_file* plus the date and hour of the run. The date and hour at the end of the file name are important to avoid accidental overwriting.

4.2 The configuration file

A really important part of SmithWaterman-CUDA is represented by the configuration file named **config.ini.** It must be in the same directory of the executable.

Through this, it is possible to control a lot of aspects of the execution of SmithWaterman-CUDA, as for example the computational resources to be used.

The configuration file is composed by different fields that the user can set to different values. Here there is the list of fields, values and their meanings.

CPU: this field takes an **F (false)** or a **T (true)** as value. Default: T. Through this, the user can decide to use (T) the CPU to work on the alignments.

CPUNUM: this field takes a positive integer as value. Default: 1. Through this, the user can set the number of CPU cores used. Obviously it doesn't make any sense to set CPUNUM=2 if ther CPU has only one core.

GPU: this field takes an **F (false)** or a **T (true)** as value. Default: T. Through this, the user can decide to use (T) the GPU to work on the alignments.

GPUNUM: this field takes a positive integer as value. Default: 1. Through this, the user can set the number of GPU used. Obviously it doesn't make any sense to set GPUNUM=2 if there is only one GPU.

MAT: it is the first algorithm-specific field. Default: BL50. Through this, the user can set the substitution matrix used. At the moment there are three possible choices: BL50 (for blosum50), BL62(for blosum62), BL90(for blosum90) and DNA1 (for identity, match = +5, mismatch = -4).

GAP_FIRST: this field takes an integer as value. Default: 10. Through this, the user can set the penalty for opening a gap.

GAP_NEXT: this field takes an integer as value. Default: 2. Through this, the user can set the penalty for extending a gap.

SCORES_THRLD: this field takes a real number as value. Default: 0. Through this, the user decides that only the alignment scores over this threshold will be saved into the output file.

SCORES_SCALING_FACTOR: this field takes an **F (false)** or a **T (true)** as value. Default: F. Through this, the user can decide to activate a kind of normalization of the output scores. In fact, sometimes it could be significant to divide the alignments scores by the one obtained aligning the query sequence with itself (this alignment obviously gives the maximum possible score).

OUTDIR: this field takes a string as value. Default: result. Through this, the user can set the output directory where to save the alignments results. The directory used for the output and whose name is given to this field must be created before running the application.

SSE2: this field may be set to **F (false, default)** or **T (true)**. When set to T it enables an SSE2 implementation on the CPU which is much faster than the common CPU implementation but it does not support COMPUTE_ENDPOSITIONS = T

COMPUTE_ENDPOSITIONS: this field may be set to **F (false, default)** or **T (true)**. When enabled it makes the software calculate endpositions of the local alignment for both the query and the subject

As said in 1, when the user choices more than one computational device (GPU or CPU), the application dynamically manages the load balancing according to their number and their computational power. The database is splitted in the same number of segments as the number of resources. Each device then computes the alignment of the query with one database segment. The size of the segment depends upon the power of that device. The speed of each resource is computed after every alignment. A new partitioning of the database is done for the successive query on the base of a weighted average of the performances detected during previous runs. Pre-fixed weights are used for the first run.

5 Troubleshooting

In this section there are some suggestions to solve some problems that can be encountered while using the application:

1. For instance, if you have a quad-core processor and two GPUs, you could try to set CPUNUM=4 and GPUNUM=2 to fully exploit your computational power. But you will see that only two cores of the CPU will be activated. This is not an error. In fact a consideration on the relationship between CPU and GPU has to be done. Each GPU used to compute alignments needs to be managed by an idle core of the CPU. Thus in the case above, two cores of the CPU manage the two GPUs and the remaining two are available for computation.

2. With sequence longer than 400 residues it is necessary to use at least a core of the CPU. Serious problems could be encountered if this warning was unheard.

3. Trying to run the application, a message like this *"NVIDIA: could not open the device file /dev/nvidiactl (No such file or directory)."* means that drivers for the NVIDIA GPUs are not installed.

4. Trying to run the application, a message like this "*error while loading shared libraries:* ...." means that the variable *LD_LIBRARY_PATH* has not been correctly set and exported. See 3.

5. The application can manage queries with a maximum lenght of 2050 residues. This is due to a limitation on the available local memory of the GPU. The user can gather all the queries above this threshold using (only for these) SmithWaterman-CUDA without any GPUs activated.

5.0.1 Double GPU

An important detail for the troubleshooting section is represented by the problem that can occur trying to **run the application with more than one GPU**.

It could happen that SmithWaterman-CUDA finds from zero to one GPU. To solve the problem try to insert two boostrap kernel parameters: *uppermem=524288* and *vmalloc=256M.*

6 The output file

As said before, SmithWaterman-CUDA computes the maximum value of the alignment between each query sequence and each sequence in the given database. All the output scores are saved in a single file but divided according to the query sequence.

The output file, that can be found into the output directory (4.2), has a name composed by the *query_sequences_file* plus the *database_file* plus the date and hour of the run. The date and hour at the end of the file name are important to avoid accidental overwriting.

Suppose that we are aligning the file *query.fasta* with the DB *uniprot.fasta* at 17.23.05 on the 19/07/2007. The output file will be created in the output directory with the name *query_uniprot_17_23_05_19_07_2007.out.* If in the query file we have two sequences (O29181 and P03630) the output file will be like in the following figure.

For each query the user can find the alignment scores with the entire DB ordered by descending order. If option COMPUTE_ENDPOSITIONS is set to true than two more columns apper in the output file: Q_END (endpoint local elignment in the query) and S_END (endpoint local elignment in the subject)

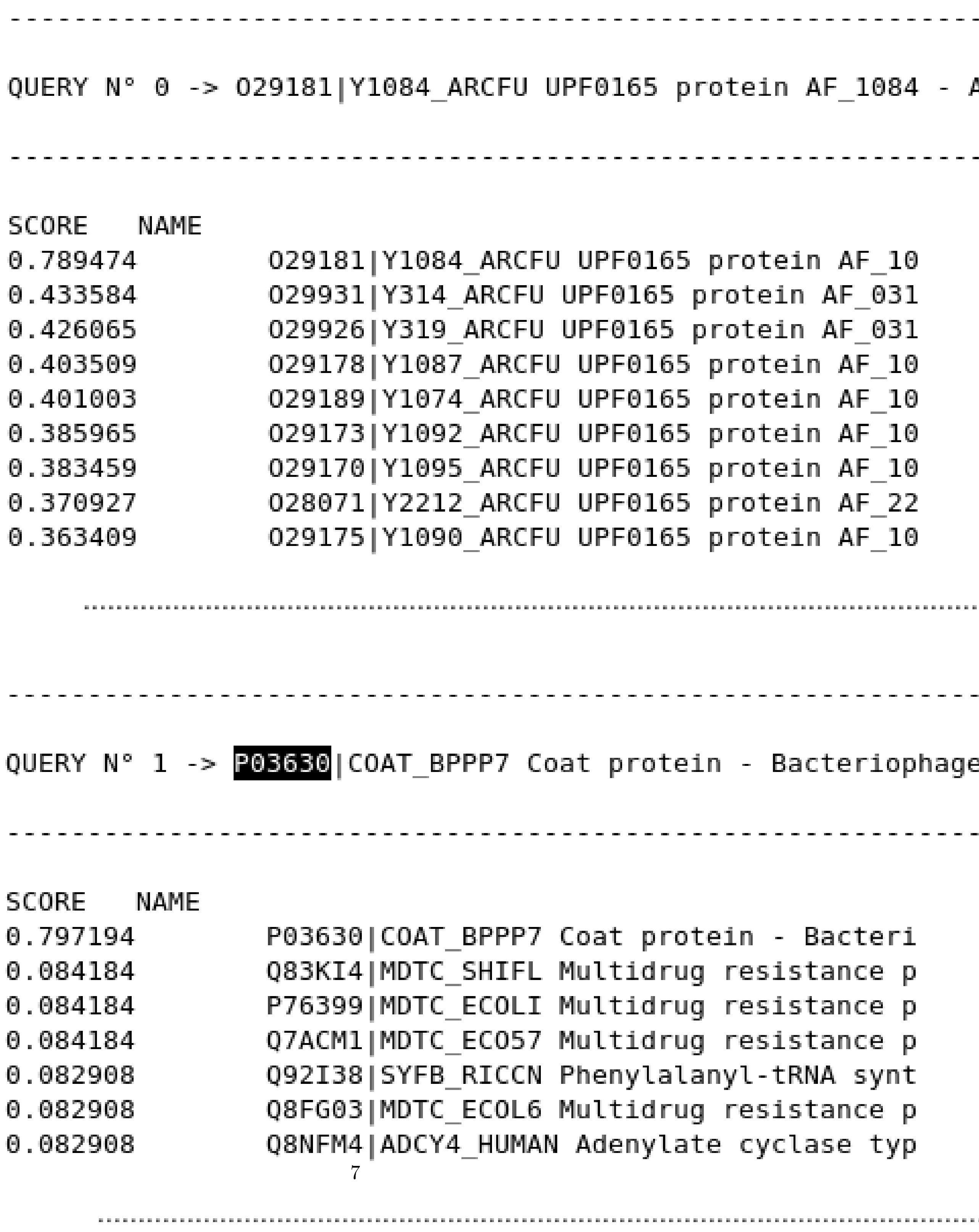
7 A simple example

In this section we describe a simple example that can guide the user while using SmithWaterman-CUDA.

Suppose, as in 6, that we have two queries (O29181 and P03630) in the file *query.fasta* and the DB in the file *uniprot.fasta.* Both the files are in the same directory of the executable. Furthermore, suppose also that we want to align only the second query using 1 core of the CPU, 1 GPU, the blosum50 matrix, an opening penalty equal to 10, an extension penalty equal to 2, an output directory called out and using the normalization of results saving only those ones above 0.08.

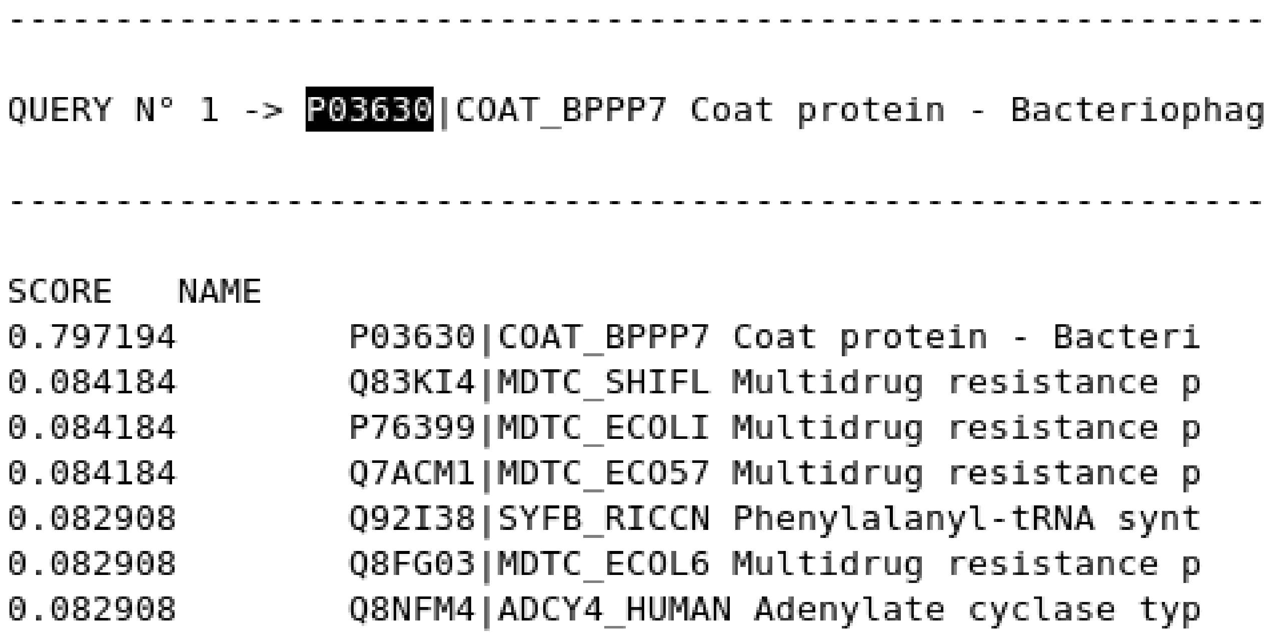
Start setting the configuration file in a proper way:

1. CPU=T
2. CPUNUM=1
3. GPU=T
4. GPUNUM=1
5. MAT=BL50
6. GAP_FIRST=10



```
---------------------------------------------------------------

QUERY N° 0 -> O29181|Y1084_ARCFU UPF0165 protein AF_1084 - A

---------------------------------------------------------------

SCORE   NAME
0.789474       O29181|Y1084_ARCFU UPF0165 protein AF_10
0.433584       O29931|Y314_ARCFU UPF0165 protein AF_031
0.426065       O29926|Y319_ARCFU UPF0165 protein AF_031
0.403509       O29178|Y1087_ARCFU UPF0165 protein AF_10
0.401003       O29189|Y1074_ARCFU UPF0165 protein AF_10
0.385965       O29173|Y1092_ARCFU UPF0165 protein AF_10
0.383459       O29170|Y1095_ARCFU UPF0165 protein AF_10
0.370927       O28071|Y2212_ARCFU UPF0165 protein AF_22
0.363409       O29175|Y1090_ARCFU UPF0165 protein AF_10

     ..........................................................

---------------------------------------------------------------

QUERY N° 1 -> P03630|COAT_BPPP7 Coat protein - Bacteriophage

---------------------------------------------------------------

SCORE   NAME
0.797194       P03630|COAT_BPPP7 Coat protein - Bacteri
0.084184       Q83KI4|MDTC_SHIFL Multidrug resistance p
0.084184       P76399|MDTC_ECOLI Multidrug resistance p
0.084184       Q7ACM1|MDTC_ECO57 Multidrug resistance p
0.082908       Q92I38|SYFB_RICCN Phenylalanyl-tRNA synt
0.082908       Q8FG03|MDTC_ECOL6 Multidrug resistance p
0.082908       Q8NFM4|ADCY4_HUMAN Adenylate cyclase typ

     ..........................................................
```

Figure 1: A typical SmithWaterman-CUDA output file.



```
-------------------------------------------------------

QUERY N° 1 -> P03630|COAT_BPPP7 Coat protein - Bacteriophag

-------------------------------------------------------

SCORE    NAME
0.797194        P03630|COAT_BPPP7 Coat protein - Bacteri
0.084184        Q83KI4|MDTC_SHIFL Multidrug resistance p
0.084184        P76399|MDTC_ECOLI Multidrug resistance p
0.084184        Q7ACM1|MDTC_ECO57 Multidrug resistance p
0.082908        Q92I38|SYFB_RICCN Phenylalanyl-tRNA synt
0.082908        Q8FG03|MDTC_ECOL6 Multidrug resistance p
0.082908        Q8NFM4|ADCY4_HUMAN Adenylate cyclase typ
```

Figure 2: The outfile in the example above.

7. GAP_NEXT=2
8. SCORE_THRLD=0.08
9. SCORE_SCALING_FACT=T
10. OUTDIR=out

To proceed and complete the alignment, use the following command ./smithwaterman *query.fasta uniprot.fasta 1.* If the run has been done at 17.23.05 on the 19/07/2007, in the directory out you can find the file *query_uniprot_17_23_05_19_07_2007.out* that looks like this:

References

[1] Bio Sequence Database Scanning on GPU, W. Liu, B. Schmidt, G. Voss, A. Schroder, W. Muller-Wittig.

[2] `http://developer.nvidia.com/object/cuda.html`.

A Installing CUDA run-time

Follow the instructions at
http://www.nvidia.com/object/cuda_get.html