GASW v1.0 user's guide

2010

Contents

Chapter 1

Introduction

GASW (short for GPU Accelerated Smith-Waterman) is the end product of an MSc. thesis on biological sequence alignment [5]; please refer to this document for detailed information on its inner workings. It is a complete graphics processing unit based implementation of the Smith-Waterman sequence alignment algorithm. GASW can be used to search FASTA format protein databases such as Swiss-Prot [3] with a recorded performance of up to around 21 GCUPS on NVIDIA GTX275.

Features:

- CUDA-based implementation of the Smith-Waterman sequence alignment algorithm for proteins, optimized for NVIDIA GT200-class GPUs.
- Aligns FASTA-format sequences with FASTA-format databases such as Swiss-Prot.
- Only returns alignment scores, not the actual alignments. However, top scoring sequences can be exported for alignment by tools that do offer this feature; see Section 2.3.
- Supports FASTA-format substitution matrices and user configurable affine gap penalties.
- Lenient limits on database and query sequence lengths, see Chapter 3.
- Comes with an easy to use web interface that allows the tool to be used remotely and with a graphical user interface, see Chapter 4.

Requirements:

- GASW is a 32-bit Microsoft Windows application; it works fine on 64 bit operating systems. It should easily compile for Unix based operating systems as it contains no platform-dependent code; however this has not been tested.
- An NVIDIA CUDA-compatible GPU with the Shader Model 1.3 feature set is required. Effectively this means GT200-based GPUs and newer. GASW should work on GT400-series GPUs, but has not been optimized for these.

Chapter 2

Program usage

This chapter explains the usage of the command-line database conversion and sequence alignment tools. Furthermore, it shows how to use a third party application to generate the actual alignments.

2.1 Converting databases to GPUDB format

Before a database can be searched using GASW, it must be converted to its own *GPUDB* format using the `dbconv` command-line program. This has to be done just once for each database. The conversion process involves separating the sequences and their descriptions and optimizing the sequence layout for GPU access. `Dbconv`'s usage is simple: its only parameter is the input database file, from which it produces `out.gpudb` (sequences) and `out.gpudb.descs` (descriptions) files. These files can be renamed; however both files for one database must have the same name and must reside in the same directory. Note that GASW comes with a small pre-converted test database named `prot_test.gpudb`, the result of converting the similarly included `prot_test.lseg` file.

Example: converting Swiss-Prot to GPUDB format

The following example shows how to convert a Swiss-Prot database to GPUDB format and how to rename it so it can be differentiated from possible other databases.

```
D:\gasw>dbconv ..\..\uniprot_sprot.fasta
Conversion parameters: block size 16, sub block size 8, alignment 256.
Loading...
..\..\uniprot_sprot.fasta: 183273162 symbols in 519348 sequence(s) in database.
Converting...
Writing out.gpudb
336 blocks
513975 sequences used to fill gaps
448 bytes of alignment padding inserted.
1822358 bytes of padding inserted.
1.033235 new vs original size ratio.
Writing out.gpudb.descs
Done.


D:\gasw>ren out.* swissprot.*

D:\gasw>dir
...
31-08-2010  16:28       191.464.944 swissprot.gpudb
31-08-2010  16:28        56.809.785 swissprot.gpudb.descs
...
```

2.2 GASW usage

Once a database has been converted to GPUDB format, the `gpu` command line program can be used to perform the actual search. Its usage is similar to that of FASTA's `ssearch` [4] program:

```
gpu [options] <sequence> <database>
```

Where *sequence* is a FASTA-format single-sequence file, and *database* is a GPUDB-format database. Additionally, the program supports the following command-line options:

Switch	Required	Description	Default
-s	yes	Substitution matrix file: FASTA-format substitution matrix	
-f	no	Gap penalty	-10
-g	no	Gap extend penalty	-2
-b	no	Number of result scores to show/export	20
-o	no	Output file for result scores (see Section 2.3)	

As noted a substitution matrix is required; GASW comes with the `blosum62.mat` Blosum62 matrix. The FASTA suite of programs offers various additional matrices in its `data` subdirectory. Once run, `gpu` will output a list of descriptions and scores for the top scoring sequences. The following example illustrates this.

Example: searching the test database

The following example shows how to search the `prot_test` database for the `mgstm1.aa` sequence. Both come with the program, as does the `blosum62` substitution matrix used.

```
D:\gasw>gpu -s blosum62.mat mgstm1.aa prot_test.gpudb
Sequence: mgstm1.aa
Database: prot_test.gpudb
Substitution matrix: blosum62.mat
Gap penalty: -10
Gap extend penalty: -2
Number of scores to show: 20
Output database file for top scoring sequences: (null)

mgstm1.aa: input sequence length is 218.
Loading database...
prot_test.gpudb: 2245 symbols in 23 sequence(s) in 1 block(s) in database.

Launching kernel.
Using 120 blocks of 64 threads: 7680 threads for 23 sequences in 1 blocks.
Processing 1 blocks per half warp.
Running...
Done. Seconds: 0.020000, GCUPS: 0.024470

Sorting results...
Results:
  0. GT8.7 | 266 | transl. of pa875.con, 19 to 675 @P:2  SCORE: 1171
  1. XURTG | 266 | glutathione transferase (EC 2.5.1.18  SCORE: 150
  2. HMIVV | 2581 | Hemagglutinin precursor - Influenza  SCORE: 38
  3. OKBO2C | 296 | Protein kinase (EC 2.7.1.37), cAMP-  SCORE: 34
  4. HAHU| 1114 | Hemoglobin alpha chain - Human, chimp  SCORE: 30
  5. RKMDS | 677 | Ribulose-bisphosphate carboxylase (E  SCORE: 27
  6. TPHUCS | 1322 | Troponin C, skeletal muscle - Huma  SCORE: 26
  7. K1HUAG | 1091 |  Ig kappa chain V-I region (Ag) -  SCORE: 25
  8. CCHU | 1 | Cytochrome c - Human @P:25-85  SCORE: 25
  9. K3HU | 1099 | Ig kappa chain C region - Human  SCORE: 24
 10. N2KF1U | 1021 | Long neurotoxin 1 - Many-banded kr  SCORE: 20
 11. FEPE | 25 | Ferredoxin - Peptostreptococcus asacch  SCORE: 19
 12. PADDING | PADDING | PADDING  SCORE: 0
 13. PADDING | PADDING | PADDING  SCORE: 0
 14. PADDING | PADDING | PADDING  SCORE: 0
 15. PADDING | PADDING | PADDING  SCORE: 0
 16. PADDING | PADDING | PADDING  SCORE: 0
 17. PADDING | PADDING | PADDING  SCORE: 0
 18. PADDING | PADDING | PADDING  SCORE: 0
 19. PADDING | PADDING | PADDING  SCORE: 0
```

The output shows the settings used; the time consumed by the alignment; and the top scoring sequences and their scores. The `PADDING` sequences are a result of the database conversion process; the original database contained just 12 sequences.

2.3 Performing alignment of top scoring sequences using SSearch

As described, GASW only calculates alignment scores and does not perform the actual sequence alignments. However, the top scoring database sequences can be exported to a new database file which can then be searched by a program that does perform full alignments. An example of such a program is

`ssearch`, which comes with the FASTA suite of programs. As the full alignments will only be performed for a relatively tiny amount of sequences, the overheard incurred by this somewhat redundant approach is negligible. If the command line used in the previous example is modified with the `-o` option, as such:

```
D:\gasw>gpu -s blosum62.mat -o out.lib mgstm1.aa prot_test.gpudb
```

The same output will be shown, however the listed sequences will be exported to the file `out.lib`. The number of sequences exported can be set using the `-b` option. The generated file can then be searched using `ssearch`. Care must be taken to provide the same options:

```
ssearch35sse2 -s blosum62.mat mgstm1.aa out.lib
...
The best scores are:                                      s-w bits E(12)
GT8.7 | 266 | transl. of pa875.con, 19 to 675 @P:2 ( 218) 1171 542.7 2.4e-158
XURTG | 266 | glutathione transferase (EC 2.5.1.18 ( 222)  150 71.8 1.4e-016
HAHU| 1114 | Hemoglobin alpha chain - Human, chimp ( 141)   30 17.2      2.2
HMIVV | 2581 | Hemagglutinin precursor - Influenza ( 567)   38 18.8      2.9
OKBO2C | 296 | Protein kinase (EC 2.7.1.37), cAMP- ( 350)   34 17.7      3.7
RKMDS | 677 | Ribulose-bisphosphate carboxylase (E ( 140)   27 15.8        5
CCHU | 1 | Cytochrome c - Human @P:25-85            ( 105)   25 15.3      5.2
K1HUAG | 1091 |  Ig kappa chain V-I region (Ag) -  ( 109)   25 15.3      5.5
K3HU | 1099 | Ig kappa chain C region - Human      ( 106)   24 14.8      6.5
TPHUCS | 1322 | Troponin C, skeletal muscle - Huma ( 159)   26 15.2      7.4
FEPE | 25 | Ferredoxin - Peptostreptococcus asacch (  54)   19 13.5      7.5
N2KF1U | 1021 | Long neurotoxin 1 - Many-banded kr (  74)   20 13.5      8.9
 More scores? [0]
 Display alignments also? (y/n) [n] y
```

Note how the same sequences and scores are shown. As the prompt for alignments was answered positively, the actual alignments are displayed:

```
...
                10         20         30         40         50         60
GT8.7   MPMILGYWNVRGLTHPIRMLLEYTDSSYDEKRYTMGDAPDFDRSQWLNEKFKLGLDFPNL
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
GT8.7   MPMILGYWNVRGLTHPIRMLLEYTDSSYDEKRYTMGDAPDFDRSQWLNEKFKLGLDFPNL
                10         20         30         40         50         60
...
```

Note that the web interface (Chapter 4) automates the task of running `ssearch` on GASW output, saving command-line work and showing the alignments in a more attractive web page format.

Chapter 3

Program limitations

Although care has been taken to be as lenient and flexible as possible, GASW is unfortunately subject to some limitations. Some are inherent to its design, some are unavoidable and some are the result of external factors. First and foremost, as it was written using the NVIDIA CUDA programming interface, it is only compatible with certain GPUs as described in Chapter 1. Furthermore, it only returns alignment scores, not the actual alignments of sequences. This problem can be circumvented by feeding the top scoring sequences into a program that does perform full alignments as discussed in Section 2.3. Additionally, due to limited development resources, GASW only supports single-GPU operation.

Finally, Microsoft Windows Vista and later implement a *timeout detection and recovery* mechanism to recover from GPU hangs [2]. By default, this mechanism resets the primary (desktop) GPU after two seconds are spent on a single task. As GASW only supports a single GPU, the primary one will always be used, which makes this timeout mechanism somewhat problematic. When using GASW to perform longer alignments, the GPU will be reset and the program aborted after two seconds. Fortunately, the timeout detection and recovery mechanism can be altered. By modifying the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\GraphicsDrivers\TdrDelay` registry entry using `regedit`, the timeout value can be set. GASW comes with a registry file `timeout.reg`; running this file will set the timeout to 255 seconds, which should be enough for most work.

Sequence limitations

The following limits are imposed on sequence (database) lengths. Note that in some cases, multiple limits apply; which one of these is the final limiting factor will depend on the particular sequences and GPU used.

Description	**Limited by**	**Limit**
Length of query sequence	GPU memory size	A temporary data matrix of size $num\_threads^1 * 4 * query\_length$ bytes is stored in global memory.
Length of query sequence	GPU memory size	A query profile of size $23 * query\_length$ bytes is stored in global memory.
Length of query sequence	Score of alignment	The maximum score supported is 65535.
Number of database sequences	GPU memory size	A score array of size $2 * num\_database\_sequences$ bytes is stored in global memory.
Number of database sequences	32-bit integer size	Disregarding memory limits, at most 4294967296 sequences are supported.
Total database size	GPU memory size	Amount of global GPU memory left after storing other data.
Total database size	GPU memory size	Disregarding other memory limits, a database size of at most 4 gigabytes is supported.
Longest sequence in database	Score of alignment	The maximum score supported is 65535.

Table 3.1: Sequence limitations

[1] *num_threads* is the amount of GPU multiprocessors times 256, for example $30 * 256 = 7680$ for GTX275.

Chapter 4

Web interface

To cut back on repetitive command-line work, especially when interested in full alignments, and to be able to show results in a more attractive manner, GUSW comes with a web interface. This interface offers all the options of the command-line program with the additions of being able to look up sequences in Swiss-Prot and the ability to invoke GUSW remotely: users can run queries from any computer, even if it does not have a powerful GPU.

4.1 Setting up the web interface

The web interface is a simple set of PHP scripts. It has been tested with the Apache 2.2.11 web server but should be compatible with any server that will run PHP scripts. Note that the web interface is not compatible with Apache running as a Windows service: services do not have access to display adapters, which prevents CUDA from being used.

Installing the web interface is a matter of copying the files from the `interface` directory of the GASW distribution to somewhere in the server's web directory. The next step is to edit `config.php` and set some parameters:

- `GPU_LOCATION` should be set to the location of GASW's `gpu.exe` file. Example: `C:/GASW/gpu.exe`.
- `SSEARCH_LOCATION` should be set to the location of the FASTA `ssearch.exe` program or one of its variants. Example: `C:/FASTA/bin/ssearch35.exe`.
- `DB_LOCATION` should be set to the location where the web interface should look for GASW GPUDB database files. Of course, `gpu.exe` should be able to access this location. Example: `C:/GASW/db`
- `MATRIX_LOCATION` should be set to the location where the web interface should look for substitution matrices. It is convenient to set this to the `data` directory of a FASTA installation. Example: `C:/FASTA/data`.

4.2 Using the web interface

Once the web interface has been properly installed, it can be accessed using any web browser. The interface index page is shown in Figure 4.1. The interface offers the following options:

- Query sequence: the query sequence to be used in the alignment, must be a file in FASTA format. The file can be anywhere on the user's system; it will be temporarily uploaded to the server hosting the web interface during the alignment.
- Database: allows the database for the alignment to be chosen. Shows a list of all databases present in the server's `DB_LOCATION` path.
- Substitution matrix: allows the substitution matrix for the alignment to be chosen. Shows a list of all matrices present in the server's `MATRIX_LOCATION` path.

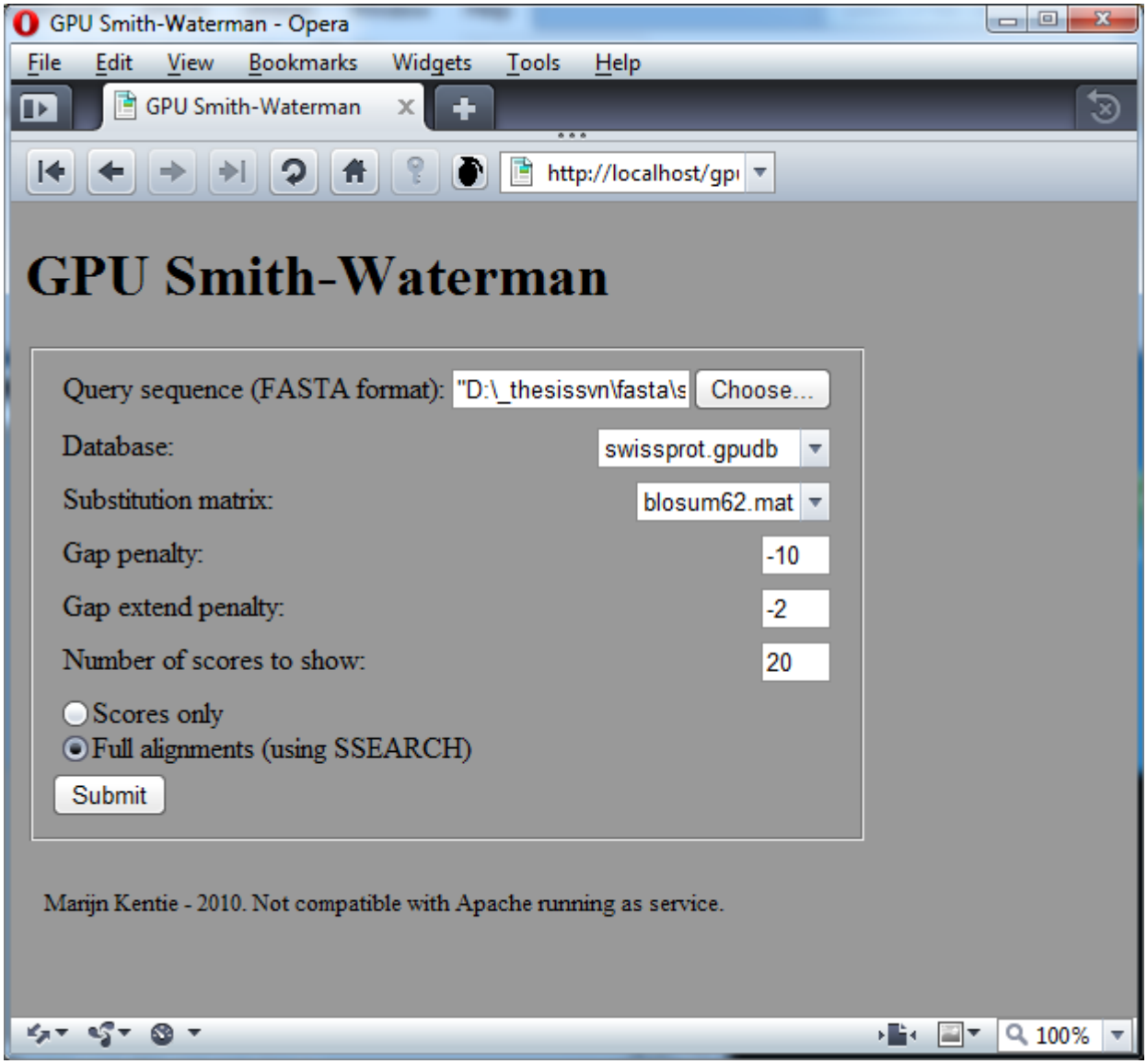

Figure 4.1: GASW web interface index page.

- Gap penalty / gap extend penalty: Smith-Waterman alignment parameters.
- Number of scores to show: The number of top scoring sequences that will be shown.
- Scores only / Full alignments. When 'Full alignments' is selected, the top scoring sequences will be run through SSearch and the actual alignments shown.

Once the query has been submitted and run, the result page shown in Figure 4.2 will be shown. The table shows the top scoring sequences and their alignment scores. The sequence identifiers are links that will show the Swiss-Prot page for the sequence as shown in Figure 4.3. If the full alignment option was used, each table entry has an additional 'alignment' link that will show the SSearch output for the sequence, see Figure 4.4.

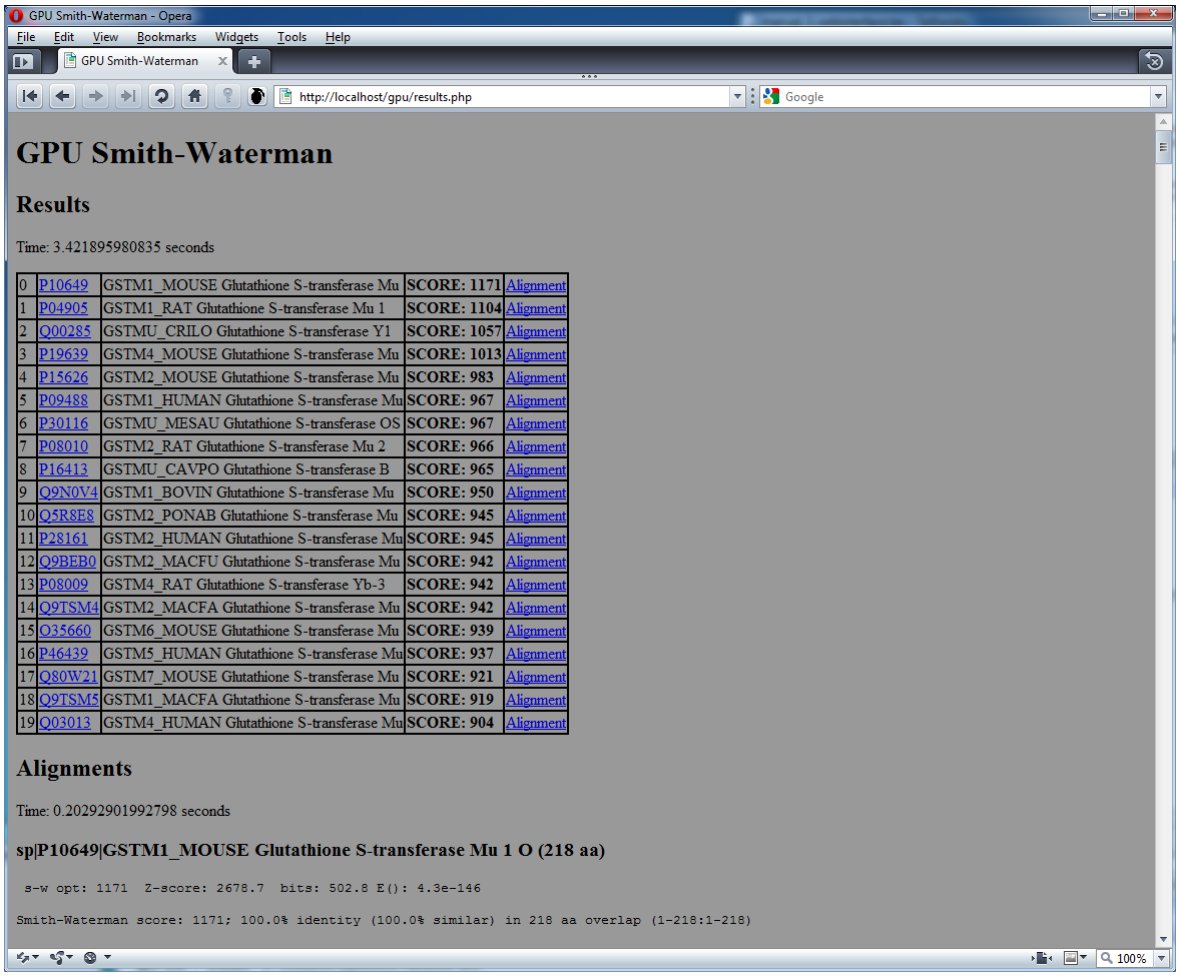

Figure 4.2: GASW web interface results page

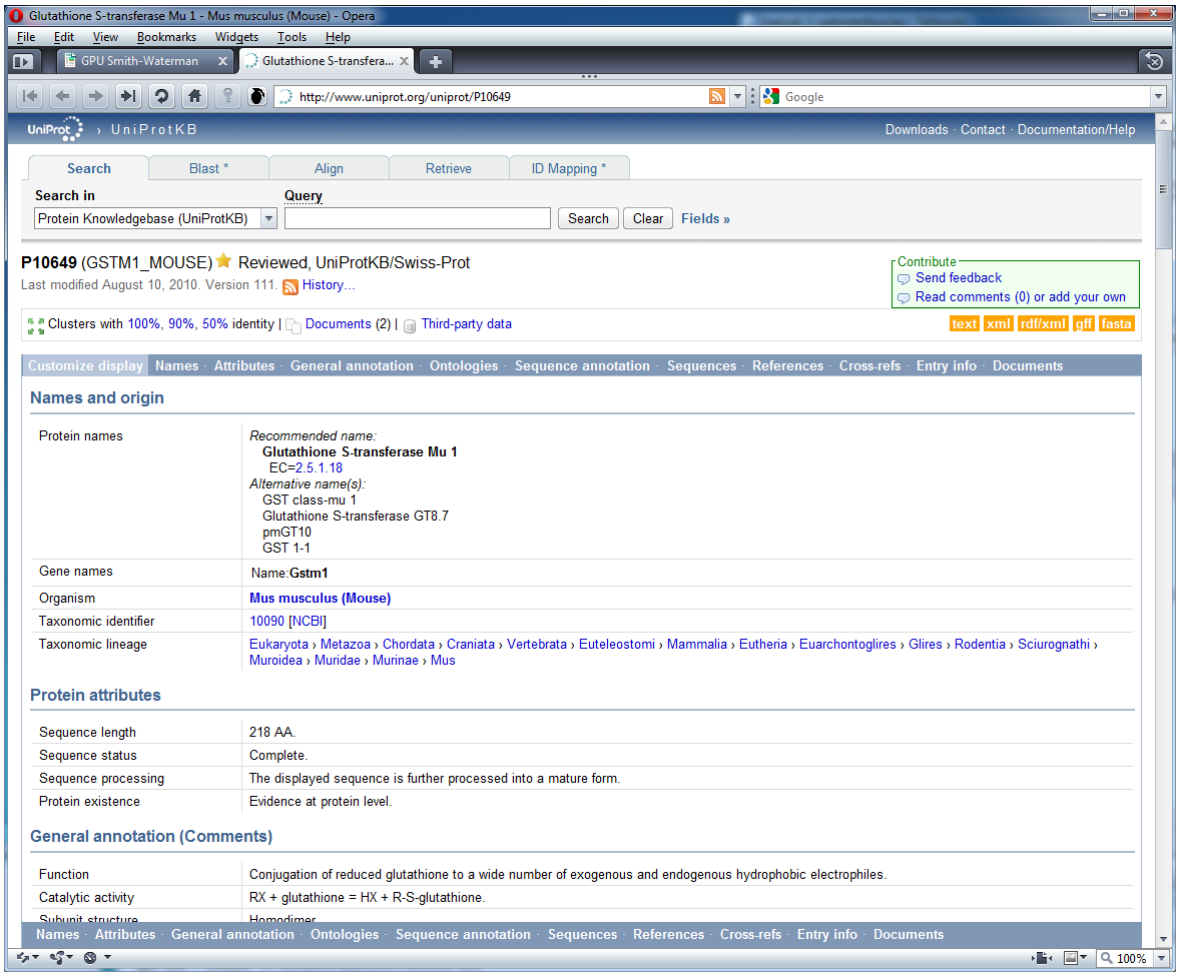

Figure 4.3: The Swiss-Prot page for the top sequence.

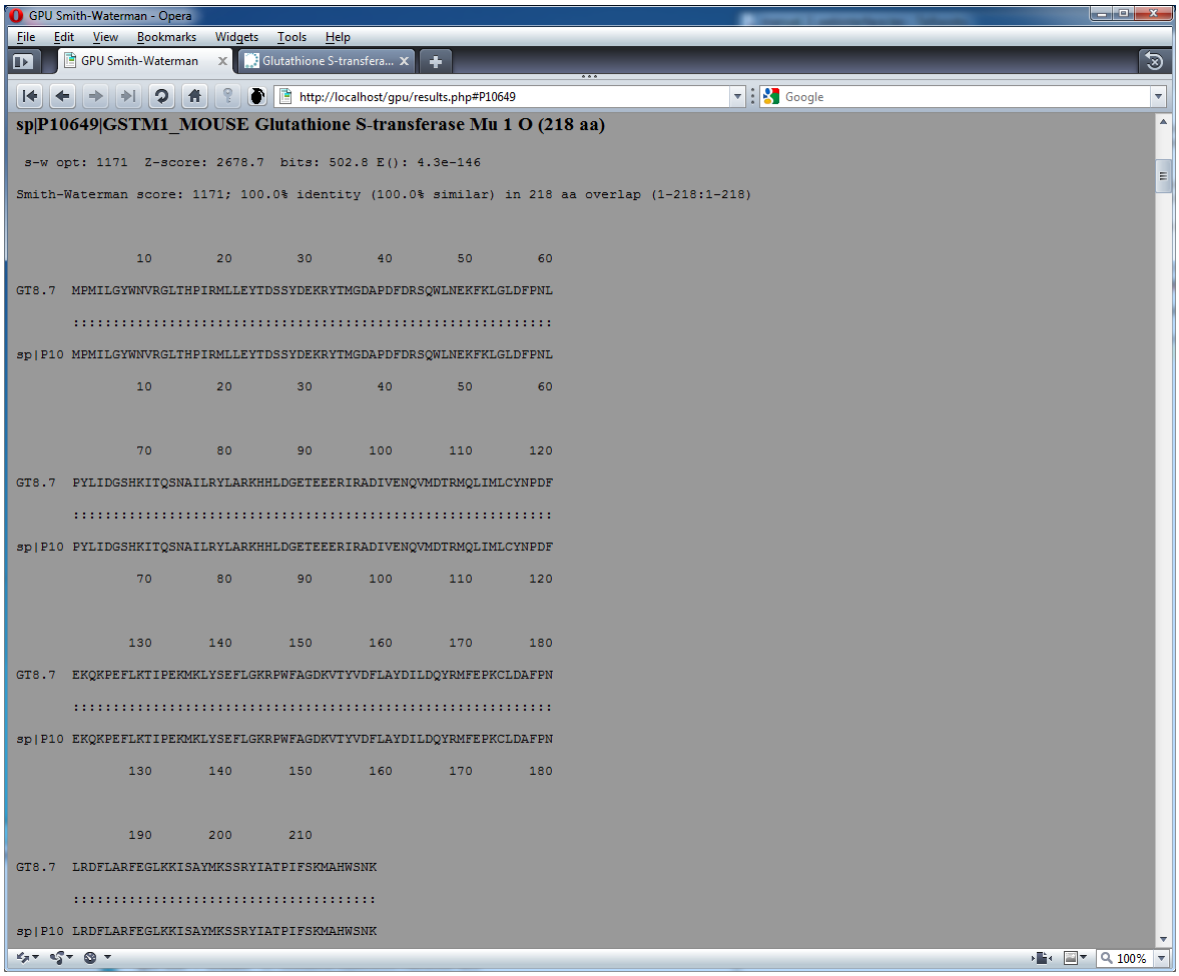

Figure 4.4: The alignment for the top sequence.

Chapter 5

Building GASW from source

GASW was originally developed using the NVIDIA CUDA toolkit version 3.1 [1]; the 32-bit version to be specific. As this version of the toolkit supports Microsoft Visual Studio 2008 at the latest, that version was used in the creation of GASW. GASW consists of a few subprojects:

- `dbconv`, the database converter that converts databases from FASTA to GASW's GPUDB format.
- `gpu`, the actual GPU accelerated Smith-Waterman implementation.
- `dbgen`, a small program to generate random FASTA databases for testing purposes.

Building `dbconv` and `dbgen` should be straightforward. To build `gpu` the CUDA toolkit must have been installed and Visual Studio must be able to find its include and library files. The `gpu` project uses the `cuda.vsprops` property sheet that has been added to the project to resolve these: the *C/C++* → *general* → *Additional Include Directories* option has been set to `$(CUDA_INC_PATH)` while the *Linker* → *Input* → *Additional Dependencies* option has been set to `$(CUDA_LIB_PATH)\cudart.lib`. These settings should work for any CUDA installation as long as the installer has set the proper environment variables. However, it is important to be aware of how the project finds these files in the event of compile errors.

The actual CUDA file in the `gpu` project, `main.cu`, must be compiled using a CUDA build rule. The CUDA 3.1 build rule comes with the source code (`cuda.rules`). If necessary, build rules can be added by right-clicking the `gpu` project in the solution explorer and selecting the *Custom Build Rules...* option. If all is well, opening the properties for `main.cu` should show the window in Figure 5.1.

As GASW is optimized for GT200-class cards, the file should be built for SM_13 architectures. Furthermore, the default 32 register limit is too low for optimal performance; 64 (or less) registers should be used with the included code compiling to use 63. When more than 64 registers are used speed will suffer due to insufficient occupancy. However, it is best to set the register limit to some higher value: this way code that compiles to more than 64 registers can be modified instead of having the compiler silently spill registers to slow local memory.

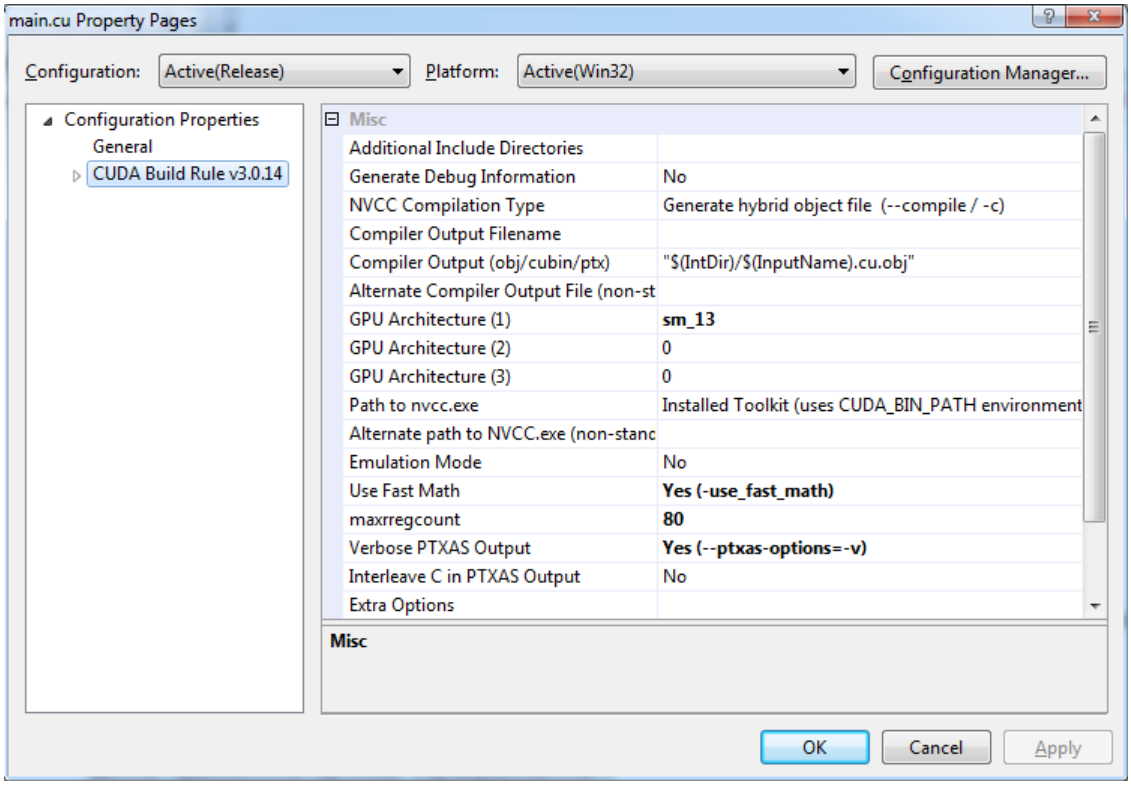

Figure 5.1: CUDA build rules property sheet.

Bibliography

[1] CUDA Toolkit 3.1 (June 2010), August 2010. http://developer.nvidia.com/object/cuda_3_1_downloads.html.

[2] Timeout Detection and Recovery of GPUs through WDDM, August 2010. http://www.microsoft.com/whdc/device/display/wddm_timeout.mspx.

[3] Universal Protein Resource, April 2010. http://www.uniprot.org.

[4] UVA FASTA Downloads, August 2010. http://fasta.bioch.virginia.edu/fasta_www2/fasta_down.shtml.

[5] M.A. Kentie. A Smith-Waterman based protein database search tool for Graphics Processing Units. 2010.