

C C A T T 0 1 0 0 0
G A G G A 0 1 1 0 1
G A A T T 0 0 1 1 0
A C A A G 0 0 1 0 0
T A C C A 0 0 1 1 0
T T A C A 0 1 0 0 0
A C C T C 0 0 0 1 0
A A G G A 0 0 0 0 0
G A T G A 0 1 1 0 0
T A G A T 0 0 1 0 0
G A T G A 1 0 1 0 0
T G T A G 1 0 0 0 0
T A G T A 0 0 0 0 0
G A T A T 1 0 0 0 0
G A G T G 1 1 0 0 0
A G A T T 1 1 0 0 0
G A G T A 1 1 0 0 0
T G A T G 1 1 0 0 0
A T T A G 1 1 0 0 0
T A G A T 1 1 0 0 0
G A G A 1 1 0 0 0
G T A 1 1 0 0 0
G A T 1 1 0 0 0
T A G 1 1 0 0 0
A G A 1 1 0 0 0
G A 1 1 0 0 0
A 1 1 0 0 0
T 1 1 0 0 0

White Paper

White paper on *CLC Bioinformatics Cell 2.1.2*

March 27, 2009



Contents

1 Introduction	4
2 Requirements	5
3 Smith-Waterman BLAST	6
3.1 BLAST vs Smith-Waterman	6
3.1.1 Sensitivity	7
3.2 Types of database searches	7
3.3 Introduction to the Smith-Waterman BLAST benchmarks	8
3.3.1 Data used in the benchmark	8
3.4 Benchmark of Smith-Waterman BLASTn	9
3.5 Benchmark of Smith-Waterman BLASTp	9
3.6 tBLASTn, BLASTx, and tBLASTx	10
4 ClustalW	14
4.1 Introduction to the ClustalW benchmarks	14
4.1.1 Data used in the benchmark	15
4.2 Benchmark of ClustalW	15
5 HMMER	16
5.1 HMMER on the Cell	17
5.2 Introduction to the HMMER benchmarks	17
5.2.1 Data used in the benchmark	18
5.3 Benchmark of hmmpfam	18
5.4 Benchmark of hmmsearch	19
5.5 Comparing the results with the original HMMER	20
5.5.1 Methods	20
5.5.2 Results	22
6 Different speeds on different CPUs	24
6.1 Higher speeds on multiple CPUs and multiple cores on the same CPU	24
6.2 How to calculate speed (GCUPS)	24

References

White Paper

1 Introduction

March, 2009: White paper updated

- We have included a comparison of the results of the original HMMER and the Cell version (see section 5.5).

CLC Bioinformatics Cell is a highly specialized combination of software and hardware for high performance bioinformatics.

It currently includes three accelerated algorithms:

- **Smith-Waterman BLAST** which combines the accuracy and sensitivity from the Smith-Waterman algorithm with the well-known input and output formats of BLAST.
- The popular **ClustalW** alignment algorithm.
- The domain searching **hmmpfam** and **hmmsearch** algorithms from the HMMER software package.

Using the CLC Bioinformatics Cell and compared with the standard software versions of the algorithms, the speed improves up to:

	No cell	3.0 Ghz P 4	2.17 Ghz Core 2 Duo	Described in
Smith-Waterman (GCUPS)	0.05	3.0	13.0	Section 3.3
ClustalW	100%	460%	1430%	Section 4.1
HMMER (GCUPS)	0.01	0.38	1.20	Section 5.2

Table 1: Speed of algorithms using the Cell.

The Cell is a software package which takes advantage of the computing powers of new CPUs not utilized by standard software. It uses the Single Instruction Multiple Data (SIMD) technique to parallelize, and thereby accelerate the above bioinformatics functionalities (read more about SIMD on <http://en.wikipedia.org/wiki/SIMD>).

The algorithms on the Cell can be accessed in three different ways:

- Through a **command line interface**. The algorithms are designed for easy integration into an existing IT infrastructure. Both the input and output formats as well as the parameters of the three algorithms are identical with the original formats. This means that you can substitute e.g. NCBI BLAST for the Cell Smith-Waterman BLAST without modifying the scripts calling BLAST.
- Through a **graphical user interface** as an integrated plug-in to one of the CLC Workbenches: CLC DNA Workbench, CLC Protein Workbench, CLC RNA Workbench and CLC Main Workbench. All the Cell's high performance algorithms are thus accessible through a user-friendly

and graphically based user interface, side-by-side with a range of other bioinformatics tools supporting common bioinformatics work flows of the user. This makes it easy for users not familiar with command line interfaces to take advantage of the speed of the Cell.

- Through a **web interface** giving easy access for multiple users in an organization. The interface is familiar to most users since it resembles NCBI BLAST and other online tools.

This white paper summarizes the most important aspects of the three algorithms regarding performance and quality.

More information can be found on <http://www.clccell.com>, and trial versions for evaluation can be requested by sending an email to support@clcbio.com.

2 Requirements

The Cell works on Intel- and AMD-based computers with Windows, Linux, or Apple Mac OS X operating systems.

The Cell has been successfully tested on the following Intel CPUs:

- The NetBurst microarchitecture:
 - Pentium 4 (670, 661, 660, 651, 650, 641, 640, 631, 630, 551, 541, 531, 524, 521)
 - Pentium D
 - Xeon (7150N, 7140M, 7140N, 7130M 7130N, 7120M, 7120N, 7110M, 7110N, 7041, 7040, 7030, 7020, 5080, 5063, 5060, 5050, 5030)
- The Pentium M microarchitecture:
 - Pentium M (780, 770, 765, 760, 755, 750, 745, 740, 735, 730, 725, 715, 705, 778, 758, 738, 718, 773, 753, 733J, 733, 723, 713)
 - Pentium Core Solo (T1400, T1300, U1500, U1400, U1300)
 - Pentium Core Duo (T2700, T2600, T2500, T2400, T2300, T2300E, L2500, L2400, L2300, U2500, U2400)
- The Core microarchitecture:
 - Pentium Core 2 Duo (E6700, E6600, E6400, E6300, E4300, T7600, T7400, T7200, T5600, T5500, L7400, L7200)
 - Pentium Core 2 Extreme (X6800, QX6700)
 - Xeon (3070, 3060, 3050, 3040, X3220, X3210, X5355, L5320, L5310, E5345, E5335, E5320, E5310, 5160, 5150, 5148 LV, 5140, 5130, 5120, 5110)

It has also been successfully tested on an AMD Athlon(™) 64 X2 Dual Core Processor 5000+.

3 Smith-Waterman BLAST

CLC Bioinformatics Cell enables accelerated database searches using the Smith-Waterman algorithm. Nucleotide-based database searches have been accelerated up to 158 times, and protein-base database searches have been accelerated up to 78 times.

This is significantly faster than other SIMD based Smith-Waterman implementations and a breakthrough for desktop-based bioinformatics as it gives researchers the possibility of running BLAST-like searches of a significantly higher quality than before.

With the Smith-Waterman BLAST search on the Cell, the NCBI BLAST (ver. 2.2.16) has been rewritten so that the database search parts of the BLAST algorithm have been replaced with the Smith-Waterman algorithm [Smith and Waterman, 1981].

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. We call it *Smith-Waterman BLAST* to emphasize the close relationship in input and output formats with normal BLAST.

The work flow in a Cell Smith-Waterman BLAST search is thus identical to a normal BLAST search:

1. The user defines the query sequence data set, 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), through the graphical user interface of CLC Workbench, or through a web interface.
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. When using the Cell through a command line interface, the output is similar to the output produced by NCBI's BLAST. When using the CLC Workbench, the output is shown as a graphical view of the alignment to the query sequence and a table summarizing the results. The web interface is able to report the result in a number of different output formats.

3.1 BLAST vs Smith-Waterman

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.

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.

The main benefit of using BLAST is speed. Without the Cell, database searches using Smith-Waterman are unacceptably slow, and Smith-Waterman is not a realistic alternative.

With the Cell however, the speed has been dramatically improved, and it has removed the great hindrance for using Smith-Waterman.

3.1.1 Sensitivity

Our comparison between BLAST and Smith-Waterman shows that running BLASTp with the default expectation value of 10 often result in only half as many hits as running a Smith-Waterman search using the Cell (see table 9).

On larger searches¹ the amount of missing hits on BLASTp is also significant. Table 9 shows an example where a Smith-Waterman based search finds 8,033 hits and BLAST finds only 6,794 hits. The difference of 1,239 hits means that the Smith-Waterman search finds approximately 18% more hits than BLAST.

BLASTing with lower E-values reduces the number of missing hits compared to using the Cell. 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.

If you have query sequence data with inherent errors, BLAST searches will often be so imprecise that the results are wrong or misleading. In this case, Smith-Waterman performs much better.

The considerable benefits of using Smith-Waterman compared to BLAST are also described in: "Sensitivity and selectivity in protein similarity searches: a comparison of Smith-Waterman in hardware to BLAST and FASTA" [Shpaer et al., 1996].

3.2 Types of database searches

These types of Smith-Waterman based BLAST searches can be run on the Cell:

Search type	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

Table 2: Types of database searches.

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

3.3 Introduction to the Smith-Waterman BLAST benchmarks

The following configurations are used in the benchmarks:

- A traditional software implementation of Smith-Waterman based searches using the NCBI BLAST 2.2.16 interface (i.e. without using the Cell).
- Smith-Waterman based BLAST search using the Cell (also with the NCBI BLAST 2.2.16 interface).
- The standard NCBI BLAST 2.2.16 as a reference (performed on the computer - not on NCBI's web site).

The tests have been performed on computers with the following CPUs:

- Intel 3.0 Ghz Pentium 4 with 1GB RAM running Fedora Core 3.
- Intel 2.13 Ghz Core 2 Duo E6400 (using both cores) with 2GB RAM running Fedora Core 6.
- AMD Athlon64 X2 5000+ with 2GB RAM running Fedora Core 6.
- 2 x Intel Xeon X3210 (Quad Core) with 3GB RAM running Fedora Core 6

In the following, the concept "speed-up" is used for describing the decrease in execution time when going from a Smith-Waterman search without a Cell to a Smith-Waterman search performed on the Cell.

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

3.3.1 Data used in the benchmark

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

Below you see the databases used to benchmark BLASTn and BLASTp, respectively.

Name	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 3: Test data - Benchmark of Smith-Waterman BLASTn.

Name	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 4: Test data - Benchmark of Smith-Waterman BLASTp

3.4 Benchmark of Smith-Waterman BLASTn

From table 7 it can be seen that the speed using Cell technology is 122-125 times faster than the 2.13 GHz Core 2 Duo using a traditional software implementation with 100 query sequences and an almost 20,000,000 nucleotides database.

The Cell technology runs at a speed of up to 13 GCUPS, compared to speeds of up to 0.1 GCUPS on the 2.13 GHz Core 2 Duo without the Cell. Compared to a fast desktop computer like the 3 Ghz Intel Pentium 4, this is **264 times faster** (tables 5 and 7).

Using the Cell on a computer with a state-of-the-art 8 core Intel CPU is up to 921 times faster than the 3 Ghz Intel Pentium 4 without the Cell (table 8).

As can be seen from table 5, the speed of the 3 GHz Pentium 4 is around 23% of the speed of the newer 2.13 GHz Core 2 Duo. This is due to the two cores (of 2.13 Ghz each) and the more modern microarchitecture of the Core 2 Duo CPU. See section 6 for more information on this.

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

Table 5 also shows that the number of search hits using the Smith-Waterman based BLASTn search is significantly higher than the number of search hits when using the traditional BLASTn. As an example, a search of 100 query sequences against a database of 1,000 sequences with an E-value of 10 results in 2,639 hits when using Smith-Waterman and only 2,443 hits when using BLASTn. The difference amounts to 8 %.

3.5 Benchmark of Smith-Waterman BLASTp

Because there are only 4 different nucleotides but 20 different amino acids, the speed of the Cell is reduced when doing SIMD-based amino acid searches compared to SIMD-based nucleotide searches. Due to the construction of desktop and laptop computers, the speed in a pure software database search is, however, not reduced significantly when performing amino acid searches.

Table 11 shows that on the 2.13 GHz Core 2 Duo, the speed using the Cell is more than 70 times higher than without the Cell. The Cell runs at a speed of up to 6.88 GCUPS, compared to speeds of up to 0.09 GCUPS on the 2.13 GHz Core 2 Duo without the Cell.

As can be seen from table 9, the speed of the 3 GHz Pentium 4 desktop is around 23% of the

speed of the newer 2.13 GHz Core 2 Duo desktop. This is due to the two cores (of 2.13 Ghz each) and the more modern microarchitecture of the Core 2 Duo CPU. See section 6 for more information on this.

Table 12 shows that running Smith-Waterman BLASTp on the Intel 8 core CPU is almost 600 times faster than the 3 Ghz Pentium 4 without the Cell (23.87 GCUPS compared to 0.04 GCUPS).

Table 9 shows that the number of search hits using the Smith-Waterman based BLASTp search is significantly higher than the number of search hits when using the traditional BLASTp. As an example, a search of 100 query sequences against a database of 1,000 sequences with an E-value of 10 results in 1,577 hits when using Smith-Waterman and only 734 hits when using BLASTp. The Smith-Waterman search thus finds 115% more hits than a normal BLASTp, and the difference is so significant that it will often make BLASTp practically useless for these types of searches.

3.6 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.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTn run locally	Cell hits	BLASTn hits	Index, BLASTn = 100
			Time	GCUPS	Time	GCUPS					
100	1000	0.01	17m 16s	0.05	19.1s	2.94	54.3x	0.53s	369	369	100
100	1000	10.00	17m 35s	0.05	21.4s	2.62	49.3x	1.27s	2639	2443	108
100	All	0.01	11h 26m 14s	0.05	11m 24s	3.03	60.1x	8.58s	7254	7244	100
100	All	10.00	11h 08m 50s	0.05	11m 35s	2.98	57.7x	12.7s	16044	15377	104

Table 5: Benchmark of Smith-Waterman **BLASTn** tested on an Intel 3.0 Ghz Pentium 4 processor.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTn run locally	Cell hits	BLASTn hits	Index, BLASTn = 100
			Time	GCUPS	Time	GCUPS					
100	All	0.01	5h 28m 46s	0.11	2m 37s	13.19	125.4x	5.21s	7254	7244	100
100	All	10.00	5h 30m 40s	0.10	2m 41s	12.84	122.8x	7.98s	16044	15377	104

Table 6: Benchmark of Smith-Waterman **BLASTn** tested on an Intel 2.13 Ghz Core 2 Duo processor using both cores.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTn run locally	Cell hits	BLASTn hits	Index, BLASTn = 100
			Time	GCUPS	Time	GCUPS					
100	All	0.01	4h 37m 08s	0.12	3m 50s	8.98	72.0x	4.72s	7254	7244	100
100	All	10.00	4h 30m 02s	0.13	3m 55s	8.81	68.8x	6.96s	16044	15377	104

Table 7: Benchmark of Smith-Waterman **BLASTn** tested on an AMD Athlon64 X2 5000+ processor using both cores.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTn run locally	Cell hits	BLASTn hits	Index, BLASTn = 100
			Time	GCUPS	Time	GCUPS					
100	All	0.01	1h 42m 42s	0.34	45.0s	46.07	136.8x	4.33s	7254	7244	100
100	All	10.00	2h 05m 40s	0.28	47.5s	43.71	158.9x	7.06s	16044	15377	104

Table 8: Benchmarks - Smith-Waterman **BLASTn** tested on an Intel 2.12 Ghz 8 core processor using all cores

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTp run locally	Cell hits	BLASTp hits	Index, BLASTp = 100
			Time	GCUPS	Time	GCUPS					
100	1000	0.01	4m 04s	0.05	8.25s	1.46	29.6x	1.90s	199	195	102
100	1000	10.00	4m 38s	0.04	10.0s	1.20	27.7x	2.37s	1577	734	215
100	50000	0.01	4h 45m 46s	0.04	6m 27s	1.60	44.3x	59.6s	4353	4277	102
100	50000	10.00	4h 53m 17s	0.04	6m 39s	1.55	44.0x	1m 11s	8033	6794	118

Table 9: Benchmarks - Smith-Waterman **BLASTp** tested on an Intel 3.0 Ghz Pentium 4 processor.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTp run locally	Cell hits	BLASTp hits	Index, BLASTp = 100
			Time	GCUPS	Time	GCUPS					
100	50000	0.01	1h 52m 23s	0.09	1m 30s	6.88	74.8x	28.0s	4353	4277	102
100	50000	10.00	1h 55m 19s	0.09	1m 35s	6.51	72.7x	35.0s	8033	6794	118

Table 10: Benchmarks - Smith-Waterman **BLASTp** tested on an Intel 2.13 Ghz Core 2 Duo processor using both cores.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTp run locally	Cell hits	BLASTp hits	Index, BLASTp = 100
			Time	GCUPS	Time	GCUPS					
100	50000	0.01	1h 38m 15s	0.11	2m 12s	4.67	44.4x	28.1s	4353	4277	102
100	50000	10.00	1h 37m 40s	0.11	2m 17s	4.52	42.8x	33.3s	8033	6794	118

Table 11: Benchmarks - Smith-Waterman **BLASTp** tested on an AMD Athlon64 X2 5000+ processor using both cores.

No of query sequences	No of seq. in database	E-value	No Cell		Cell		Speed-up	NCBI BLASTp run locally	Cell hits	BLASTp hits	Index, BLASTp = 100
			Time	GCUPS	Time	GCUPS					
100	50000	0.01	33m 46s	0.31	26.0s	23.87	78.0x	11.2s	4353	4277	102
100	50000	10.00	31m 58s	0.32	28.2s	21.96	68.0x	15.2s	8033	6794	118

Table 12: Benchmarks - Smith-Waterman **BLASTp** tested on an Intel 2.12 Ghz 8 core processor using all cores

4 ClustalW

The ClustalW alignment algorithm [Thompson et al., 1994] is probably the most popular algorithm used for making multiple alignments. On the Cell, ClustalW has been accelerated up to 31 times (see table 17).

The command line version of ClustalW included in the *CLC Bioinformatics Cell* produces the same output and uses the same input as the standard ClustalW software.

This means that you can readily insert the accelerated Cell version of ClustalW into an existing setup of scripts and tools without further adjustments. It also means that you don't have to learn to use a new tool for doing alignments.

In the plug-in version of ClustalW, there is a graphical interface to both the input and the output, making it easy for to use for those not familiar with command line tools. Furthermore, there is the benefit of the advanced graphical viewing and editing capabilities of the CLC Workbench, making it possible to e.g. display annotations from the input sequences on the alignment.

4.1 Introduction to the ClustalW benchmarks

The following configurations are used in the benchmarks:

- The standard implementation of ClustalW obtained from <ftp://ftp.ebi.ac.uk/pub/software/clustalw2>.
- The Cell version of ClustalW.

The tests have been performed on computers with the following CPUs:

- Intel 3.0 Ghz Pentium 4 with 1GB RAM running Fedora Core 3.
- Intel 2.13 Ghz Core 2 Duo E6400 (using both cores) with 2GB RAM running Fedora Core 6.
- AMD Athlon64 X2 5000+ with 2GB RAM running Fedora Core 6.
- 2 x Intel Xeon X3210 (Quad Core) with 3GB RAM running Fedora Core 6

In the following, the concept "speed-up" is used for describing the decrease in execution time when going from the standard ClustalW algorithm to the Cell version of ClustalW.

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

If you compare the result of the Cell version of ClustalW with the original version, there might be some minor differences. They can occur if two pairwise alignments score equally well, and the Cell version chooses another alignment than ClustalW. This, however does not affect the quality of the alignment. Another reason for small differences is due to a minor error in the original ClustalW which we have corrected with the version 2.0 Cell.

4.1.1 Data used in the benchmark

The test data used for the ClustalW benchmarks are shown in table 4.1.1.

Name in benchmark	Number of sequences	Average length	Type
Globin	744	144.2	Protein
Gyrase-A	60	869.6	Protein
HIV-1	62	9014.9	Nucleotide

Table 13: Test data - Benchmark of ClustalW.

The *Globin* data set is the complete hemoglobin alpha, beta, gamma, delta, and epsilon sequences from Swiss-Prot.

The *Gyrase-A* data set are the complete *Gyrase subunit A* sequences from Swiss-Prot.

The HIV-1 data set are sequences from the 2005 HIV-1 subtype reference alignment. Retrieved from the HIV database at Los Alamos National Laboratory http://www.hiv.lanl.gov/content/hiv-db/SUBTYPE_REF/align.html.

4.2 Benchmark of ClustalW

On a standard Desktop computer as the Intel 3.0 Ghz Pentium 4 processor, the speed-up of ClustalW is about 4 times, so that alignments taking more than 8 minutes without the Cell are reduced to less than 2 minutes (see table 14).

When looking at the CPUs with multiple cores, the speed increases because the normal ClustalW only uses one core. With the Intel 2.13 Ghz Core 2 Duo, standard ClustalW alignments taking 3 hours and 52 minutes are performed in 22 minutes using the Cell. This amounts to a speed-up of 10 (see table 15). The benchmarks of the AMD Athlon64 X2 5000+ processor show a similar trend with speed-ups up to 11 times (see table 16).

On the Intel 2.12 Ghz 8 core, the speed-up is even more dramatic: up to 31 times the standard ClustalW making it possible to perform alignments of 62 sequence with an average length of 9015 in just 7 minutes and 18 seconds (see table 17).

Thus, the greatest increase in speed of ClustalW is observed for CPUs with multiple cores. The Intel 2.13 Ghz Core 2 Duo processor with the Cell sets a new standard for the size of the alignments that can be performed on a new, standard desktop computer.

Data set	No of sequences	Average length	No cell	Cell	Speed-up
Globin	744	144.2	8m 32s	1m 50s	4.6x
Gyrase-A	60	869.6	1m 59s	36.3s	3.3x
HIV-1	62	9014.9	4h 15m 17s	1h 13m 50s	3.5x

Table 14: Benchmark of **ClustalW** tested on an Intel 3.0 Ghz Pentium 4 processor.

Data set	No of sequences	Average length	No cell	Cell	Speed-up
Globin	744	144.2	7m 56s	33.4s	14.3x
Gyrase-A	60	869.6	1m 52s	11.2s	10.0x
HIV-1	62	9014.9	3h 52m 49s	21m 55s	10.6x

Table 15: Benchmark of **ClustalW** tested on an Intel 2.13 Ghz Core 2 Duo processor using both cores.

Data set	No of sequences	Average length	No cell	Cell	Speed-up
Globin	744	144.2	7m 57s	42.2s	11.3x
Gyrase-A	60	869.6	1m 53s	15.1s	7.5x
HIV-1	62	9014.9	4h 05m 12s	36m 54s	6.6x

Table 16: Benchmark of **ClustalW** tested on an AMD Athlon64 X2 5000+ processor using both cores

Data set	No of sequences	Average length	No cell	Cell	Speed-up
Globin	744	144.2	7m 55s	18.1s	26.4x
Gyrase-A	60	869.6	1m 51s	5.01s	22.3x
HIV-1	62	9014.9	3h 51m 56s	7m 18s	31.7x

Table 17: Benchmark of **ClustalW** tested on an Intel 2.12 Ghz 8 core processor using all cores

5 HMMER

Profile hidden Markov models (profile HMMs) are used to do sensitive database searching using statistical descriptions of a sequence family's consensus. HMMER is an implementation of profile HMM software for protein sequence analysis. The *CLC Bioinformatics Cell* includes an accelerated version of HMMER with a speed-up of up to 32 times.

When doing databases searches like BLAST and Smith-Waterman searches, they are based on pairwise alignments between the query sequences and the sequences in the database. HMMER uses profile HMMs instead of pairwise alignments. This means that a hit is found based on the information of many sequences instead of just one sequence.

The Pfam database at <http://www.sanger.ac.uk/Software/Pfam/> is a large collection of multiple sequence alignments that covers approximately 8000 protein domains and protein families [Bateman et al., 2004]. Based on the individual domain alignments, profile HMMs have been developed. These profile HMMs can be used to search for domains in unknown sequences using HMMER.

Many proteins have a unique combination of domains which can be responsible, for instance, for the catalytic activities of enzymes. Pfam was initially developed to aid the annotation of the *C. elegans* genome.

The Pfam database can be used to search for domains in sequence data which otherwise do not carry any annotation information.

It is the two main search programs in the HMMER package, *hmmpfam* and *hmmsearch*, which have been accelerated on the Cell. *hmmpfam* is used to find domains in one or more sequences by searching a database of profile HMMs. *hmmsearch* is used to search a sequence database to find matches to a specific profile HMM.

You can read more about HMMER at <http://hmmmer.janelia.org/>.

5.1 HMMER on the Cell

When using the command line version of HMMER, it produces the almost the same output² and uses the same input as the standard HMMER package.

This means that you can readily insert the accelerated Cell version of HMMER into an existing setup of scripts and tools without further adjustments.

When using HMMER as a plug-in to the CLC Workbench, a variety of sequence formats can be used. The output comes in three different versions: as the standard text output similar to the command line, as annotations on the sequences, in a table format providing overview of the results and making it easy to copy and paste into e.g. Excel.

Next follows a benchmark of the Cell version of HMMER focused on performance, and after that we make some detailed comparisons of the output of the original HMMER and the Cell version.

5.2 Introduction to the HMMER benchmarks

The following configurations are used in the benchmarks:

- The standard implementation of HMMER obtained from <http://hmmmer.janelia.org/>.
- The Cell version of HMMER.

Note that it is the two search algorithms, *hmmpfam* and *hmmsearch*, which have been accelerated on the Cell. The other, less frequently used, programs in the HMMER package remain unchanged.

The tests have been performed on computers with the following CPUs:

- Intel 3.0 Ghz Pentium 4 with 1GB RAM running Fedora Core 3.
- Intel 2.13 Ghz Core 2 Duo E6400 (using both cores) with 2GB RAM running Fedora Core 6.
- AMD Athlon64 X2 5000+ with 2GB RAM running Fedora Core 6.
- 2 x Intel Xeon X3210 (Quad Core) with 3GB RAM running Fedora Core 6

²only, the histogram is omitted from the output of *hmmsearch*

In the following, the concept "speed-up" is used for describing the decrease in execution time when going from the standard HMMER algorithm to the Cell version of HMMER.

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

5.2.1 Data used in the benchmark

The following data has been used in the benchmark of hmmpfam and hmmsearch, respectively.

Database	HMMs/sequences	Positions
100 proteins from Swiss-Prot	100	40,993
Pfam 21.0 fs + ls	17,914	3,772,912

Table 18: Test data - Benchmark of hmmpfam

Database	HMMs/sequences	Positions
Swiss-Prot 53.3	274,295	100,696,439
10 models from Pfam fs + ls	20	3,176

Table 19: Test data - Benchmark of hmmsearch

A binary version of the Pfam database is used. This reduces the amount of time spent reading from the disk, since the binary database format is more compact than the text format.

5.3 Benchmark of hmmpfam

On a standard Desktop computer as the Intel 3.0 Ghz Pentium 4 processor, the speed-up of hmmpfam is 24-28 times, so that searches taking more than 3 hours without the Cell are reduced to less than 8 minutes using the Cell (see table 20).

When looking at the CPUs with two cores, the speed-up follows the same trend. With the Intel 2.13 Ghz Core 2 Duo, the speed-up is 30-36 times, and with the AMD Athlon64 X2 5000+ the speed-up is 28-32 times (see tables 21 and 22).

With the Intel 2.12 Ghz 8 core, speed-up is 28-35 times (see table 23). T

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
17914	100	0.01	3h 13m 54s	0.01	6m 47s	0.38	28.5x
17914	100	10.00	3h 13m 53s	0.01	7m 50s	0.33	24.7x

Table 20: Benchmark of **hmmpfam** tested on an Intel 3.0 Ghz Pentium 4 processor.

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
17914	100	0.01	1h 18m 45s	0.03	2m 09s	1.20	36.6x
17914	100	10.00	1h 16m 58s	0.03	2m 33s	1.01	30.1x

Table 21: Benchmark of **hmmpfam** tested on an Intel 2.13 Ghz Core 2 Duo processor using both cores.

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
17914	100	0.01	1h 29m 03s	0.03	2m 45s	0.94	32.3x
17914	100	10.00	1h 28m 52s	0.03	3m 09s	0.82	28.1x

Table 22: Benchmarks - **hmmpfam** tested on an AMD Athlon64 X2 5000+ using both cores

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
17914	100	0.01	20m 20s	0.13	34.3s	4.50	35.5x
17914	100	10.00	20m 21s	0.13	43.5s	3.56	28.1x

Table 23: Benchmark of **hmmpfam** tested on an Intel 2.12 Ghz 8 core processor using all cores.

5.4 Benchmark of hmsearch

On a standard Desktop computer as the Intel 3.0 Ghz Pentium 4 processor, the speed-up of hmsearch is 17.6 times, so that searches taking more than 3 hours without the Cell are reduced to about $11\frac{1}{2}$ minutes using the Cell (see table 24).

When looking at the Intel 2.13 Ghz Core 2 Duo, the speed-up is 29.2 which is due to the modern architecture of this CPU compared with the Intel 3.0 Ghz Pentium 4. Searches taking more than $1\frac{1}{2}$ hours without the Cell are performed in merely 3 minutes and 13 seconds using the Cell (see table 25).

The AMD Athlon64 X2 5000+ is not quite as fast with a speed-up of about 23 times (26).

With the Intel 2.12 Ghz 8 core, speed-up is over 23 times (see table 27), and search time is reduced to 1 minute. Compared with the Intel 3.0 Ghz Pentium 4 without the Cell using more than 3 hours, this is more than 175 times faster!

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
20	274295	0.01	3h 21m 17s	0.03	11m 26s	0.47	17.6x
20	274295	10.00	3h 21m 14s	0.03	11m 27s	0.47	17.6x

Table 24: Benchmark of **hmmsearch** tested on an Intel 3.0 Ghz Pentium 4 processor.

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
20	274295	0.01	1h 33m 55s	0.06	3m 13s	1.66	29.2x
20	274295	10.00	1h 33m 55s	0.06	3m 13s	1.65	29.2x

Table 25: Benchmark of **hmmsearch** tested on an Intel 2.13 Ghz Core 2 Duo processor using both cores.

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
20	274295	0.01	1h 41m 57s	0.05	4m 25s	1.20	23.0x
20	274295	10.00	1h 43m 08s	0.05	4m 23s	1.22	23.5x

Table 26: Benchmarks - **hmmsearch** tested on an AMD Athlon64 X2 5000+ using both cores

No of pro-files	No of seq. in database	E-value	No cell		Cell		Speed-up
			Time	GCUPS	Time	GCUPS	
20	274295	0.01	23m 57s	0.22	1m 00s	5.29	23.8x
20	274295	10.00	23m 46s	0.22	1m 00s	5.32	23.7x

Table 27: Benchmark of **hmmsearch** tested on an Intel 2.12 Ghz 8 core processor using all cores.

5.5 Comparing the results with the original HMMER

Since the Cell version of HMMER is a completely new implementation, we have made some comparisons of the result with the original HMMER.

5.5.1 Methods

The data sets used were the Pfam database (version 23.0) and the Swiss-Prot database (version 15.0). A random subset of each of these was created, containing 500 HMMs and 2,000 proteins,

respectively.

The `hmmsearch` versions were compared by using the subset of 500 HMMs to search the full Swiss-Prot database containing 428,650 proteins. The `hmmpfam` versions were compared by using the subset of 2,000 proteins and characterizing them according to the full Pfam database of 10,340 HMMs. For each comparison, two runs were made, one using the `fs` HMMs (modeling domain fragments) and one using the `ls` HMMs (modeling full domains).

The tests were run on a Linux machine with dual quad core 2.50 GHz Intel Xeon E5420 CPUs, i.e. the machine contained eight CPU cores. The machine has 16 GB of RAM but the programs used far less.

The HMMER version used was 2.3.2 with threading enabled. All runs were done using default parameters. The original HMMER runs were done using binary HMM files to minimize the effect of parsing time.

For each comparison of an HMM to the Swiss-Prot database and each comparison of a protein to the Pfam database, the output of the CLC Cell was compared to the output of the original HMMER software. The comparison was done on the part of the output containing the overall scores and the individual domain parsing. The actual alignments were not compared, just the placing of the domains. If a domain score and domain position is identical, then the actual alignment is bound to be very close to identical.

As described above, the CLC Cell version of `hmmsearch` does not output a histogram of the scores, so that difference is ignored in the following. Also, different amounts of spaces and the width of the description output field is ignored in the following.

The differences in output is divided into several types, ranging from the least severe to the most severe:

No difference There is no difference in output

Output order The hits are reported in different order but everything else is identical. This may happen when two hits score equally and the sorting differs.

Domain position At least one domain has a difference in the position of one of its ends in the HMM and/or protein. No differences in scores are present.

Small score difference At least one hit has a difference of at most one in the last reported decimal place of the score or e-value.

Larger score difference At least one hit has a difference of more than one in the last reported decimal place of the score or e-value.

Other difference A more substantial difference is present.

When two outputs are compared and multiple differences found, the most severe category is chosen. For example, if the output has a difference in the position of one domain and another domain has a score difference of one unit in the last decimal place, the category is "Small score difference". Note that the differences of all types, possibly apart from the last two types, are very minor and most likely due to small rounding differences.

5.5.2 Results

Below are the results of the output comparison, presented in tables containing the number of each type of difference, along with the cumulative counts. All the cases of “larger score difference” are differences of two in the last decimal places of the score or e-value. The one instance of an “other difference”, was some repeat proteins that had 10 domain matches in the original HMMER and 11 domain matches in the CLC version. The 11'th domain matches were weak enough that the overall e-value was unchanged.

hmmsearch, domain fragment (fs) HMMs

Difference type	Counts	Cumulative
No difference	307 (61.4 %)	307 (61.4 %)
Output order	79 (15.8 %)	386 (77.2 %)
Domain position	14 (2.8 %)	400 (80.0 %)
Small score difference	98 (19.6 %)	498 (99.6 %)
Larger score difference	1 (0.2 %)	499 (99.8 %)
Other difference	1 (0.2 %)	500 (100.0 %)

hmmsearch, full domain (ls) HMMs

Difference type	Counts	Cumulative
No difference	375 (75.0 %)	375 (75.0 %)
Output order	46 (9.2 %)	421 (84.2 %)
Domain position	0 (0.0 %)	421 (84.2 %)
Small score difference	78 (15.6 %)	499 (99.8 %)
Larger score difference	1 (0.2 %)	500 (100.0 %)
Other difference	0 (0.0 %)	500 (100.0 %)

hmmpfam, domain fragment (fs) HMMs

Difference type	Counts	Cumulative
No difference	1,419 (71.0 %)	1,419 (71.0 %)
Output order	438 (21.9 %)	1,857 (92.9 %)
Domain position	24 (1.2 %)	1,881 (94.1 %)
Small score difference	119 (6.0 %)	2,000 (100.0 %)
Larger score difference	0 (0.0 %)	2,000 (100.0 %)
Other difference	0 (0.0 %)	2,000 (100.0 %)

hmmpfam, full domain (ls) HMMs

Difference type	Counts	Cumulative
No difference	1,521 (76.1 %)	1,521 (76.1 %)
Output order	365 (18.3 %)	1,886 (94.3 %)
Domain position	1 (0.1 %)	1,887 (94.4 %)
Small score difference	111 (5.6 %)	1,998 (99.9 %)
Larger score difference	2 (0.1 %)	2,000 (100.0 %)
Other difference	0 (0.0 %)	2,000 (100.0 %)

It is evident from the results that in at least 99.5 % of the runs of each type, the result differences between the CLC Cell and the original HMMER are extremely small. The remaining runs also have very inconsequential differences. In all of the 5,000 runs, the reported sequences and HMMs that matched were identical.

6 Different speeds on different CPUs

As the Cell utilizes the architecture of the computer's CPU, the Cell runs at different speeds on different CPUs.

One parameter is the clock frequency and another is the CPU's microarchitecture.

The CPUs relevant for using the Cell can basically be categorized in three types of CPU microarchitectures:

1. The NetBurst microarchitecture including, among others, the Pentium 4 and Pentium D processors
2. The Pentium M microarchitecture including, among others, the Pentium M and Pentium Core Solo processors
3. The Core microarchitecture including, among others, the Intel Core 2 Duo and Intel Core 2 Extreme processors

The above is listed in the order of speed when using the Cell technology (see section 2 for a comprehensive list of supported CPUs). At a given clock frequency, the speed using the Cell technology is typically more than twice as high on the Core microarchitecture than on the NetBurst microarchitecture.

The Intel Core 2 Duo and Intel Core 2 Extreme processors were released mid 2006, and most top-line Intel-based computers produced since then contain one or more of these processors.

Within the same type of microarchitecture, the speed using the Cell technology is approximately linearly dependent on the clock frequency. A 3 GHz NetBurst processor is thus approximately twice as fast as another type of NetBurst Processor with a clock frequency of 1.5 GHz.

On the exact same type of CPU, the speed using the Cell technology is linearly dependent on the clock frequency.

6.1 Higher speeds on multiple CPUs and multiple cores on the same CPU

Many modern computers have two CPUs or multi-core CPUs.

A multi-core CPU combines two or more independent processors into a single package. A dual-core CPU contains two independent microprocessors and a quadcore CPU contains four independent microprocessors.

The Smith-Waterman algorithm can be parallelized on multiple CPUs and multiple cores as shown in the benchmark tables 5 to 12.

6.2 How to calculate speed (GCUPS)

The Smith-Waterman search algorithm compares all nucleotides/amino acids in the query sequence(s) with all nucleotides/amino acids in the database.

Based on this concept, the speed of a Smith-Waterman database search can be calculated as the number of comparisons per second.

The speed of the Smith-Waterman BLASTp 100 vs. 1,000 search in table 9 is thus calculated as

$$\frac{33,701 \times 357,564}{9.81s} = 1,228,365,378 \text{ comparisons per second} \sim 1.23 \text{ (GCUPS)}$$

When calculating speed for Smith-Waterman BLASTn, the number of cell updates is twice that of Smith-Waterman BLASTp because the output is based on DNA/RNA sequence comparisons made both forward and backward.

The speed of the Smith-Waterman BLASTn 100 vs. 1,000 search in table 5 above is thus calculated as ³

$$\frac{52,489 \times 533,828}{20.5s} \times 2 = 2,733,668,087 \text{ comparisons per second} \sim 2.73 \text{ (GCUPS)}$$

³Table 5 says 2.74 GCUPS whereas this example says 2.73. This is because the time with only one decimal in this example is less precise than the calculations shown in the table.

References

- [Bateman et al., 2004] Bateman, A., Coin, L., Durbin, R., Finn, R. D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E. L. L., Studholme, D. J., Yeats, C., and Eddy, S. R. (2004). The Pfam protein families database. *Nucleic Acids Res*, 32(Database issue):D138–D141.
- [Shpaer et al., 1996] Shpaer, E. G., Robinson, M., Yee, D., Candlin, J. D., Mines, R., and Hunkapiller, T. (1996). Sensitivity and selectivity in protein similarity searches: a comparison of smith-waterman in hardware to blast and fasta. *Genomics*, 38(2):179–191.
- [Smith and Waterman, 1981] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197.
- [Thompson et al., 1994] Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–4680.