1 Introduction

DIAMOND is a sequence aligner for protein and translated DNA searches and functions as a drop-in replacement for the NCBI BLAST software tools. It is suitable for protein-protein search as well as DNA-protein search on short reads and longer sequences including contigs and assemblies, providing a speedup of BLAST ranging up to x20,000.

Support & updates: http://github.com/bbuchfink/diamond

Publication: If you use DIAMOND in published research, please cite:
"Fast and sensitive protein alignment using DIAMOND", *Nature Methods* **12**, 59-60 (2015)

2 Quick start guide

We assume to have a protein database file in FASTA format named `nr.faa` and a file of DNA reads that we want to align named `reads.fna`.

In order to set up a reference database for DIAMOND, the `makedb` command needs to be executed with the following command line:

```
$ diamond makedb --in nr.faa -d nr
```

This will create a binary DIAMOND database file with the specified name (`nr.dmnd`). The alignment task may then be initiated using the `blastx` command like this:

```
$ diamond blastx -d nr -q reads.fna -o matches.m8
```

The output file here is specified with the `-o` option and named `matches.m8`. By default, it is generated in BLAST tabular format.

Note:

- The program may use quite a lot of memory and also temporary disk space. Should the program fail due to running out of either one, you need to set a lower value for the block size parameter `-b` (see 6.5).
- The default (fast) mode was mainly designed for short reads. For longer sequences, the sensitive modes (options `--sensitive` or `--more-sensitive`) are recommended.
- **The full speed of the program is only attained on large query datasets. It is currently not efficient in mapping a smaller number of query sequences.**

3 Installation & Requirements

DIAMOND runs on x86-64 compatible hardware and POSIX-compatible operating systems (Linux, FreeBSD, OS X) as well as on Microsoft Windows.

A high-memory server is recommended for better performance, but the program can be run on standard desktop computers.

DIAMOND is actively developed software, so it is recommended to always use the latest version of the program.

Compiled binaries are available for download for Linux and Windows. Users of Max OS X need to compile the software from source.

3.1 Binary download

3.1.1 Linux

A precompiled binary is available for recent Linux systems and may be downloaded for immediate use:

```
wget http://github.com/bbuchfink/diamond/releases/download/v0.8.21/diamond-linux64.tar.gz
tar xzf diamond-linux64.tar.gz
```

If the binary does not work on your system, you need to compile the software from source.

3.1.2 FreeBSD

On FreeBSD, you can use `pkg install diamond` to install the software.

3.1.3 Windows

A binary executable for Windows can be downloaded at the DIAMOND github page. You also need to install the Visual C++ Redistributable (Link).

3.2 Compiling from source

Compilation requires GCC 4.1 or later, CMake 2.6 or later as well as libpthread and zlib including development headers. To compile DIAMOND from source, invoke the following commands on the shell:

```
wget http://github.com/bbuchfink/diamond/archive/v0.8.21.tar.gz
tar xzf v0.8.21.tar.gz
cd diamond-0.8.21
mkdir bin
cd bin
cmake ..
make install
```

Note:

- Use `cmake -DCMAKE_INSTALL_PREFIX=...` to install to a different prefix.
- Use `cmake -DBUILD_STATIC=ON` to create a statically linked executable.

4 Guidelines

You should take the time to configure the program to your needs instead of just running it with default settings, as this will substantially optimize the use of system resources. Some points to consider:

- Set the -e parameter (maximum expected value) as low as possible.
- Set the -k parameter (number of target sequences to report alignments for) as low as possible. This will improve performance and reduce the use of temporary disk space and the size of the output files.
- Consider using the --top parameter instead of -k. For example, setting --top 5 will report all alignments whose score is at most 5% lower than the top alignment score for a query.
- Configure the tabular output format to only report the output fields that are actually required (see 6.2.2).

- Use the BLAST tabular format for machine-based processing of the output. Use the XML and SAM format only if required by your downstream tools or if you prefer it as a human-readable format.
- Use the option --compress 1 to automatically generate gzip-compressed output files.

5 Commands

Commands are issued as the first parameter on the command line and set the task to be run by the program.

Command	Description
makedb	Create DIAMOND formatted reference database from a FASTA input file.
blastp	Align protein query sequences against a protein reference database.
blastx	Align translated DNA query sequences against a protein reference database.
view	Generate formatted output from DAA files.

6 Options

6.1 Makedb options

Option	Short	Default	Description
--in			Path to protein reference database file in FASTA format (may be gzip compressed).
--db	-d		Path to DIAMOND database file.

6.2 General & IO options

Option	Short	Default	Description
--threads	-p	max	Number of CPU threads.
--db	-d		Path to DIAMOND database file.
--query	-q		Path to query input file in FASTA or FASTQ format (may be gzip compressed).
--out	-o	stdout	Path to output file.
--outfmt	-f	6	Format of output file (see below).
--compress		0	Compression for output file (0=none, 1=gzip).
--salltitles			Include full length subject titles in output. This option only applies to the DAA, SAM and BLAST XML format.

6.2.1 Output formats

Format ID	Description
5	BLAST XML
6	BLAST tabular
100	DIAMOND alignment archive (DAA)
101	SAM

6.2.2 Tabular fields

Format ID 6 (BLAST tabular) may be followed by a space-separated list of the following keywords, each specifying a field of the output. The default is `6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore`.

Keyword	Description

qseqid	Query Seq - id
qlen	Query sequence length
sseqid	Subject Seq - id
sallseqid	All subject Seq - id(s), separated by a ';'
slen	Subject sequence length
qstart	Start of alignment in query
qend	End of alignment in query
sstart	Start of alignment in subject
send	End of alignment in subject
qseq	Aligned part of query sequence
sseq	Aligned part of subject sequence
evalue	Expect value
bitscore	Bit score
score	Raw score
length	Alignment length
pident	Percentage of identical matches
nident	Number of identical matches
mismatch	Number of mismatches
positive	Number of positive - scoring matches
gapopen	Number of gap openings
gaps	Total number of gaps
ppos	Percentage of positive - scoring matches
qframe	Query frame
stitle	Subject Title
salltitles	All Subject Title(s), separated by a '<>'
qcovhsp	Query Coverage Per HSP

6.3 Sensitivity & speed options

Option	Short	Default	Description
--sensitive			Trigger the sensitive alignment mode with a 16x9 seed shape configuration.
--more-sensitive			This mode provides some additional sensitivity compared to the sensitive mode.
--band		auto	Dynamic programming band for seed extension. This corresponds to the maximum length of gaps that can be found in alignments.

6.4 Scoring & reporting options

Option	Short	Default	Description
--gapopen		11	Gap open penalty.
--gapextend		1	Gap extension penalty.
--matrix		BLOSUM62	Scoring matrix.
--seg			Enable SEG masking of low complexity segments in the query (yes/no). The default is no for blastp and yes for blastx.
--max-target-seqs	-k	25	The maximum number of target sequences per query to keep alignments for.
--top			Keep alignments within the given percentage range of the top alignment score for a query (overrides –max-target-seqs option).

--evalue	-e	0.001	Maximum expected value to keep an alignment.
--min-score			Minimum bit score to keep an alignment. Setting this option will override the --evalue parameter.
--query-cover			Report only alignments above the given percentage of query cover.

6.4.1 Scoring matrices

Matrix	Supported values for (gap open)/(gap extend)
BLOSUM45	(10-13)/3; (12-16)/2; (16-19)/1
BLOSUM50	(9-13)/3; (12-16)/2; (15-19)/1
BLOSUM62	(6-11)/2; (9-13)/1
BLOSUM80	(6-9)/2; 13/2; 25/2; (9-11)/1
BLOSUM90	(6-9)/2; (9-11)/1
PAM250	(11-15)/3; (13-17)/2; (17-21)/1
PAM70	(6-8)/2; (9-11)/1
PAM30	(5-7)/2; (8-10)/1

6.5 Memory & performance options

Option	Short	Default	Description
--block-size	-b	2.0	Block size in billions of sequence letters to be processed at a time.
--tmpdir	-t		Directory to be used for temporary storage.
--index-chunks	-c	4	The number of chunks for processing the seed index.

The `--block-size/-b` parameter is the main option for controlling the program's memory usage. Bigger numbers will increase the use of memory *and* temporary disk space, but also improve performance. The program can be expected to roughly use six times this number of memory (in GB). So for the default value of `-b2.0`, the memory usage will be about 12 GB.

The temporary directory is set to the output directory by default. The amount of disk space that will be used depends on the program's settings and your data. As a general rule you should ensure that 100 GB of disk space are available here. If you run the program in a cluster environment, and disk space is only available over a slow network based file system, you may want to set the `--tmpdir` option to `/dev/shm`. This will keep temporary information in memory and thus increase the program's memory usage substantially.

The `--index-chunks/-c` option can be additionally used to tune the performance. It is recommended to set this to 1 on a high memory server, which will increase performance and memory usage, but not the usage of temporary disk space.

6.6 View options

Option	Short	Default	Description
--daa	-a		Path to input file in DAA format.
--out	-o	stdout	Path to output file.
--outfmt	-f	6	Format of output file.
--compress		0	Compression for output file (0=none, 1=gzip).