Report Number: WUCSE-2011-56

2011

Mercury BLASTN Biosequence Similarity Search System: Technical Reference Guide

Authors: Jeremy Buhler

This guide documents the operation of the Mercury BLASTN system for hardware-accelerated DNA similarity search. It includes detailed information on the syntax and limitations of the system's component commands, as well as a description of the system's hardware platform suitable for administrators who need to maintain a Mercury BLASTN system. Mercury BLASTN is a product of the High Performance COmputational Biology Group at Washington University.

2011-56

Mercury BLASTN Biosequence Similarity Search System: Technical Reference Guide

Authors: Jeremy Buhler

Corresponding Author: jbuhler@wustl.edu

Web Page: http://hpcb.wustl.edu

Abstract: This guide documents the operation of the Mercury BLASTN system for hardware-accelerated DNA similarity search. It includes detailed information on the syntax and limitations of the system's component commands, as well as a description of the system's hardware platform suitable for administrators who need to maintain a Mercury BLASTN system.

Mercury BLASTN is a product of the High Performance COmputational Biology Group at Washington University.

Type of Report: Other

 High Performance Computational Biology Group, Washington University in St. Louis, MO

Mercury BLASTN Biosequence Similarity Search System: Technical Reference Guide

Version 1.01, 5/16/2011

1 Introduction

The Mercury BLASTN application is a hybrid hardware-software implementation of DNA-to-DNA sequence comparison. It presents an interface similar to that of NCBI BLASTN 2.2 (and indeed uses much of that package's source code internally) but has many changes "under the hood" to support very large sequence comparisons. Much of Mercury BLASTN's speedup comes from effective use of custom-designed hardware circuits implemented on field-programmable gate arrays (FPGAs) that accelerate parts of the BLASTN computation. Additionally, the software part of the system is heavily multithreaded to enable parallel computation across multiple CPU cores.

This reference guide gives an overview of Mercury BLASTN's organization and documents the key commands and other information needed to perform BLASTN analyses with it. Basic use of Mercury BLASTN involves steps similar to NCBI's command-line `formatdb` and `blastall` tools. However, if you plan to use the same data set as a query with multiple databases, you can take additional steps to make your searches even faster.

This guide assumes that the reader is familiar with the basic operation of command-line NCBI BLAST and with FASTA-formatted sequences.

1.1 Quick Start: Performing a Mercury BLASTN Search

The following quick-start sequence of steps should get you up and running with Mercury BLASTN immediately. For more complete documentation of the Mercury BLASTN commands, including important tips to optimize performance and sensitivity for very large comparisons, please see Section 3.

Suppose you have FASTA-formatted files `query.fna` and `db.fna`, each containing one or more DNA sequences, and you want to compare them using Mercury BLASTN. Here is the minimal set of steps needed to accomplish the comparison, assuming these files are in your current working directory. (You can find small example `.fna` files with the indicated names in `$HGBLAST_HOME/examples`.)

1. Convert your database file, `db.fna`, to a BLAST database, using the standard NCBI BLAST `formatdb` command:

   ```
   formatdb -p f -i db.fna -n db
   ```

   This step creates BLAST database files `db.n*`. (You can find a copy of NCBI's `formatdb` on your Mercury BLAST server in the directory `$HGBLAST_HOME/bin`.)

2. Log into your Mercury BLASTN server. We'll assume that the Mercury BLASTN commands are already in your search path on this machine; if not, add `$HGBLAST_HOME/bin` to your path.

3. Convert your database file into a *Mercury BLAST* database, which is stored alongside the NCBI database, using Mercury Blast's `HgFormatdb` command:

   ```
   HgFormatdb -p f -i db.fna -o db
   ```

   This step creates Mercury BLAST database files `db.mbd` and `db.mbx`.

4. Run Mercury BLASTN on your query and database. We'll assume an E-value threshold of 10^{-5}:

```
HgBlastall -p blastn -i query.fna -d db -e 1e-5 -o results.txt
```

This command creates an output file `results.txt` with the results of the comparison. The output is formatted as for NCBI BLAST's `-m 8` option, i.e. tabular output. If the output filename is omitted, the results will be piped to `stdout`.

1.2 How to Get Help

To report problems or request help using your Mercury BLASTN system, please send an email to *hgblast-help@seas.wustl.edu*.

1.3 How to Cite Mercury BLASTN

If you use Mercury BLASTN in your published research, we ask that you please include an appropriate citation for our system. Citations for Mercury BLASTN are as follows:

J. Buhler, J. Lancaster, A. Jacob, and R. Chamberlain. "Mercury BLASTN: fast streaming DNA sequence comparison." *Proc. 2007 Reconfigurable Systems Summer Institute* (RSSI), 2007.

P. Krishnamurthy, J. Buhler, R. Chamberlain, M. Franklin, K. Gyang, and J. Lancaster. "Biosequence similarity search on the Mercury system." *J. VLSI Signal Processing*, 49:101-121, 2007.

J. Lancaster, J. Buhler, and R. Chamberlain. "Acceleration of ungapped extension in Mercury BLAST." *Microprocessors and Microsystems* 33:281-9, 2009.

A citation for the Exegy FPGA infrastructure on which Mercury BLASTN runs is:

B. Brodie, R. Chamberlain, B. Shands, and J. White. "Dynamic Reconfigurable Computing." *Proc. 9th Military and Aerospace Programmable Logic Devices Int'l. Conf.* (MAPLD), 2006.

1.4 Acknowledgements

The Mercury BLASTN system was made possible by the following organizations:

- The National Institutes of Health (award R44 HG003225), which funded both development of Mercury BLASTN and free availability of our system to selected users;
- Exegy Corporation, which designed and built the specialized FPGA cards and the driver infrastructure necessary to use them;
- The High-Performance Computational Biology Group at Washington University in St. Louis, which designed and implemented the Mercury BLASTN architecture on top of Exegy's hardware platform.

2 Server Configuration

This section documents the hardware and software configuration of the Mercury BLASTN server. The information herein is principally of interest to the system administrator(s) responsible for maintaining the server.

2.1 Hardware Configuration and Requirements

The Mercury BLASTN server contains two 2.6-GHz six-core AMD Opteron processors and 32 GB of RAM, as well as two PCI-X cards containing the FPGA hardware. The server contains a 137 GB boot drive, which holds the OS, plus roughly 4 TB of usable RAID0 storage, which holds the Mercury BLAST installation and provides space for local mirrors of data files.

The server is housed in a SuperMicro 3U rack-mount chassis, which requires three 120V or three 220V redundant power connections. The system's maximum power draw is 800 W, though the usual draw is under 500W. The server requires standard machine room air, 75F max inlet temperature and, 50% max humidity. Connectivity is via gigabit Ethernet.

2.2 System Software

The Mercury BLASTN server runs CentOS Linux 5.5 x86_64 (equivalent to Redhat Enterprise ES5.5) with a custom kernel based on version 2.6.32-3. Please note that the custom kernel is required to drive the FPGA cards; a stock kernel will not work.

The partitions on the boot drive, `/dev/sda`, are formatted as ext3, while the RAID, which is visible as device `/dev/md0` and is mounted as `/s0`, is formatted as xfs. For information on administering an xfs filesystem, please see http://xfs.org/. Please note that `/s0` is writable only by root; the `/s0/scratch` subdirectory is globally writable and setgid, and other such directories may be created as desired.

2.3 Boot-Time Configuration

The Mercury BLASTN server starts two daemons when it boots: `exegyd`, which initializes and controls the FPGA cards, and `exegy-logger`, which records debugging data. These two daemons *must* be running for Mercury BLASTN to work correctly.

The init scripts that manage these daemons and perform other required setup, such as loading kernel modules and changing system defaults, may be found in `/etc/rc.d/init.d`. They include `exegyd`, `exegy-logger`, and `textminer`.

2.4 Mercury BLASTN Installation

The Mercury BLASTN software may be found in the directory `/s0/hgblast/stable`, which is a link to the current stable version of the tools. This location is exported to all users via the environment variable HGBLAST_HOME, set in the default login profiles in `/etc`.

The $HGBLAST_HOME directory contains the following subdirectories:

bin/ – Mercury BLASTN executables. This directory should be in users' search paths.

doc/ – a copy of this documentation.

examples/ -- example data for BLAST runs described in this documentation.

The directory `/s0/scratch` is the default location for temporary file creation during Mercury BLASTN runs. An alternate location may be chosen by changing the environment variable HGBLAST_TEMP (which is also set by default in `/etc`). However, it is strongly recommended that temporary file creation be confined to the local RAID `/s0` for performance reasons.

3 Mercury BLASTN Command Reference

Mercury BLASTN presents a user interface similar to that of NCBI BLASTN. For basic searches, the usage model should be familiar to any BLAST user: given FASTA files containing query and database sequences, first prepare the database sequence for searches, using the `HgFormatdb` command, and then compare the query to the database using the `HgBlastall` command. The usage of these two commands is described in Section 3.1.

More advanced usage scenarios are possible for users who want to save time or improve output quality. These alternative usages are detailed in Section 3.2.

3.1 Basic Usage

3.1.1 HgFormatdb: Preparing a Database for Mercury BLASTN Searches

Just as NCBI BLAST must preprocess a sequence database with the `formatdb` command before it can be used for searches, Mercury BLASTN requires its own preprocessing command, `HgFormatdb`. This command, which must be *run in addition to* NCBI's `formatdb`, rewrites the database into a special form that is sent to the FPGA hardware devices during a search.

Syntax

```
HgFormatdb [-p t/F] [-U t/F] -i <input file> [-o <output file>]
```

-i	name of input multi-FASTA file (required)
-o	base name of output database files
-p	true for protein, false for DNA; default is false
-U	hard-mask lower-case characters; default is false

`HgFormatdb` reads a text file consisting of one or more FASTA-formatted sequences and outputs a Mercury BLAST-formatted database. The database consists of two files with extensions .mbd and .mbx. If the user species '-o filename', then the outputs are written to `filename.{mbd,mbx}`; otherwise, the output files use the input filename with .mbd and .mbx appended.

The input sequences may use a mix of upper- and lower-case characters. By default, input case is ignored (and is not preserved); however, if '–U t' is specified, all lower-case characters will be *hard-masked*, i.e. replaced with the special character 'X'. This option is useful for, e.g., creating databases from repeat-masked genomic DNA sequence in which repeats are stored as lower-case.

Caveat: Mercury BLASTN does *not* currently support creation or use of multi-volume databases. Trying to use an NCBI-created multi-volume database with Mercury BLASTN will not yield the expected results, even if you create a single large Mercury-format database from the same input using `HgFormatdb`. Also, due to the limitations of NCBI BLAST, single database volumes containing more than 4 billion characters are not supported. If you need to search a database with more characters than this, please split the sequences into multiple files of at most 4 billion characters and prepare NCBI and Mercury-formatted databases separately for each file.

3.1.2 HgBlastall: Running a BLAST Search

The `HgBlastall` command implements the core BLASTN search algorithm, using the server's hardware acceleration to speed the computation. To perform an `HgBlastall` search, you must supply a

FASTA file containing one or more *query* sequences as well as a database, which must be preprocessed with *both* `formatdb` and `HgFormatdb`.

Please note that not all parameters understood by NCBI's `blastall` are accepted by `HgBlastall`. A list of accepted parameters is given in the command syntax below.

Syntax

```
HgBlastall -p blastn -i <query> -d <database> [-o <output file>]
      [-e <evalue>][-b <nseqs>] [-v <naligns>] [-z <size>] [-Y <size>]
      [-m <format>] [-q <score>] [-r <score>] [-G <score>]
      [-E <score>] [-X <score>] [-F T/f] [-S 1|3]
      [-U t/F] [-version] [-preserveqy <dirname>]
      [-preservedb <dirname>] [-preserverun <dirname>]
```

-p	program name (required); only "`blastn`" is currently supported
-i	FASTA query file (required)
-d	database name (required); preprocessed files name.n*, name.mbd, and name.mbx must exist
-o	output file name; default is stdout
-e	expectation value threshold (real-valued default = 1.0)
-b	max number of database sequences allowed to align to one query; default = 250
-v	max number of alignments for which summaries are shown in long-form output mode (e.g. -m 0); default = 250
-z	effective database size for Karlin-Altschul statistics (real-valued); default automatic
-Y	effective search size for Karlin-Altschul statistics (real-valued); default automatic
-m	output format; NCBI options 0, 8, and 9 are supported; default = 8
-q	nucleotide mismatch penalty; default = -3
-r	nucleotide match bonus; default = +1
-G	gap opening penalty (-1 = "default behavior"); default = -1
-E	gap extension penalty (-1 = "default behavior"); default = -1
-F	perform low-complexity filtering of query (Dust); default is true
-S	search forward (1) or both (3) strands of query; default = 3
-U	hard-mask lower-case characters in query; default is false
-version	Print Mercury BLASTN version information
-preserveqy	preserve/use preserved query directory (see Section 3.2)
-preservedb	preserve/use preserved database directory (see Section 3.2)
-preserverun	preserve/use preserved search output (see Section 3.2)

For efficiency reasons, we recommend that the query, database, and output locations all be on the Mercury BLASTN server's RAID (/s0). However, the search will work, albeit more slowly, if these locations are on remotely-mounted filesystems.

Caveats: while Mercury BLASTN supports more permissive scoring parameters for –q and –r than the defaults, parameters that generate much larger numbers of high-scoring segment pairs (HSPs) than the

defaults may trigger the system's internal guards against excessive output due to a hardware fault. If you encounter difficulty using your preferred score parameters, please contact the developers.

Currently, NCBI BLAST's -S 2 (search the reverse-complement strand of the query only) option is not supported. Support will be added in a future version of Mercury BLASTN.

The only supported output formats for alignments at this time are NCBI BLAST's –m 0 (alignments), –m 8 (tabular), and –m 9 (tabular with header) formats. –m 0 is the typical "long form" output produced by command-line NCBI BLAST. For the –m 8 and 9 options, summary information for each alignment is printed on one line in tab-separated form. The fields on each line are as follows:

1. name of the query sequence;
2. name of the database sequence;
3. percent identity of alignment;
4. number of matches in alignment;
5. number of mismatches in alignment;
6. number of gaps in alignment;
7. starting coordinate of alignment in query sequence;
8. ending coordinate of alignment in query sequence;
9. starting coordinate of alignment in database sequence;
10. ending coordinate of alignment in database sequence;
11. E-value of alignment;
12. bit score of alignment.

Alignments involving a given query are sorted in descending order by E-value, with ties broken by database position, alignment length, and query position.

3.2 Advanced Usage

To maintain an interface similar to NCBI's `blastall` while efficiently processing very large queries, the `HgBlastall` command actually runs a pipeline of several steps "under the hood." Understanding this pipeline exposes a number of opportunities to tweak Mercury BLASTN's behavior for greater speed or sensitivity.

Figure 1illustrates the detailed `HgBlastall` computation. Before a FASTA query can be used for search, it is first subject to *preprocessing*. The principal role of query preprocessing is to divide a large query into small chunks, or subqueries, each of which is searched independently against the database. Long sequences, such as an entire chromosome, are broken into overlapping subqueries, while large numbers of short sequences, such as next-generation reads, may be packed into a smaller number of multiplexed subqueries. In addition to subdividing the query, preprocessing also applies customary operations such as masking low-complexity sequence. All these steps are performed by a tool called `HgPreparequery`.

Each preprocessed subquery passes through two search stages. The first stage submits the subquery to the FPGA hardware, which compares it against the Mercury-formatted database. The output of this stage roughly corresponds to NCBI BLASTN's ungapped HSPs. The results of the hardware stage are written to disk, then passed to a modified version of NCBI `blastall` that performs gapped extension (using the NCBI-formatted database), significance estimation, and output formatting. This second search stage produces a set of zero or more alignment results for each subquery.

The last step in the HgBlastall computation is to reassemble the subquery results into a single output file. This step is performed by the HgMFAPost tool. HgMFAPost does not merely concatenate and sort the results from each subquery; it also "undoes" any splitting performed by HgPreparequery, so that the start and end coordinates of alignments in the final output represent locations in the original query sequences.

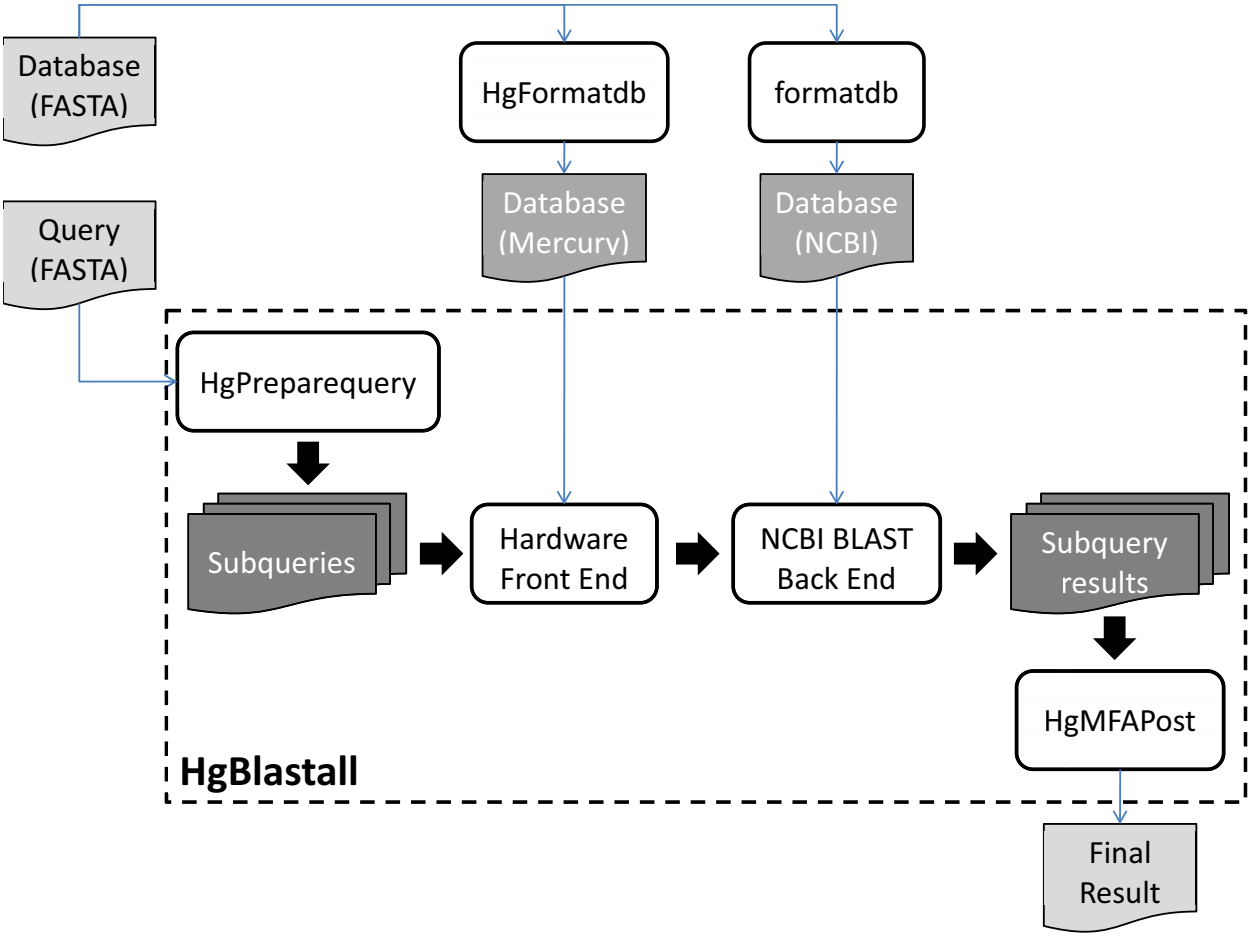

Figure 1: Internal organization of HgBlastall and supporting tools.

With the proper command-line options, it is possible to execute the various parts of the HgBlastall computation separately, capturing the intermediate results for later analysis or reuse. The following sections describe several such advanced usage scenarios and document the operation of the HgPreparequery and HgMFAPost tools, which may be run separately from HgBlastall.

3.2.1 Capturing and Reusing Preprocessed Subqueries

For queries with millions or billions of characters, the computational overhead of subquery generation in HgBlastall is substantial. If you know that you will be using the same query to search more than one database, it is more efficient to generate the preprocessed subqueries once and reuse them for all searches.

The –preserveqy parameter to HgBlastall provides access to the subqueries created as part of a search pipeline. Adding the option '-preserveqy' to a search, and specifying a *nonexistent* directory name <dirname> as its argument, will cause the preprocessed subqueries to be saved to <dirname>. To use

preprocessed subqueries in a directory <dirname> for a subsequent `HgBlastall`, simply supply the '–preserveqy' option with <dirname> as the argument in addition to the original FASTA file.

If a full path is not specified, <dirname> is assumed to be a subdirectory of `/s0/scratch/qy`.

Example

```
HgBlastall –p blastn –i query.fna –d db –preserveqy savedqy
      –o res
```

Assuming that `/s0/scratch/savedqy` *does not* previously exist, the above command runs a search of FASTA query file `query.fna` against database `db`, saving the subqueries to the directory `/s0/scratch/savedqy`.

```
HgBlastall –p blastn –i query.fna –preserveqy /s0/scratch/savedqy
      –d db2 –o res2
```

Assuming that `/s0/scratch/savedqy` *does* exist, the above command runs a search with the saved subqueries against the database `db2`.

Caveat: When using preprocessed subqueries with `HgBlastall`, *it is the user's responsibility to specify the same query-related options (-S, -U, and –F) to `HgBlastall` as were specified when the subqueries were created.* Failure to use the same options may cause undesirable behavior.

Caveat: it is highly recommended that preprocessed subqueries always be written to and read from a directory on the Mercury BLASTN server's local RAID (`/s0`). Placing subqueries elsewhere will work but will likely be much slower.

3.2.1.1 The HgPrepareQuery Command

Instead of using an `HgBlastall` search to capture preprocessed subqueries, it is possible to perform preprocessing manually using the `HgPreparequery` command.

Syntax

```
HgPreparequery [-p t/F] [-S 1|3] [-U t/F] [-F T/f]
      -d <dirname> <input file>
```

-p	true for protein, false for DNA; default is false
-S	DNA strands to process: forward (1) or both (3); default = 3
-U	hard-mask lower-case characters; default is false
-F	filter out low-complexity sequences; default is true
-d	directory that will hold the preprocessed query files (required)

`HgPreparequery` takes a file of FASTA-formatted sequences and preprocesses them to prepare them for searching with Mercury BLASTN. It writes the preprocessed subqueries into a specified directory (which should not already exist). The -S, -U, and -F parameters are interpreted as for `HgBlastall`.

3.2.2 Capturing and Reusing Local Database Files

The `HgBlastall` program can accept as its –d argument a database on any filesystem visible to the user. However, to ensure high performance, it is important that this database reside on the disk local to the accelerator hardware. Mercury BLASTN therefore copies the input database to a directory under /s0/scratch before it runs. Moreover, `HgBlastall` generates and uses some internal files from the database as part of its operation, to speed up software processing of hits discovered by the hardware, and stores these files in the same directory as its local copy of the database. The copying and internal file generation operations are transparent to the user but can add significantly to the program's running time.

If it is known that a given database will be used for multiple searches, the user may add the –preservedb option to the `HgBlastall` command line. As for –preserveqy, providing a *nonexistent* directory name <dirname> as the argument will cause the local database copy to be saved in <dirname>, while specifying a *preexisting* directory will read a previously saved database copy from that location, avoiding the overhead of copying and internal file generation. Note that –preservedb must be specified in addition to the –d option that points to the original source of the database.

If <dirname> is not an absolute path, it is assumed to be a subdirectory of `/s0/scratch/db`.

Example

The following example runs `HgBlastall` on two successive queries. If `/s0/scratch/db/saveddb` does not previously exist, the first command creates it and caches the database, while the second reuses the cached database files.

```
HgBlastall -p blastn -i query1.fna -d db -preservedb saveddb
      -o res

HgBlastall -p blastn -i query2.fna -d db -preservedb saveddb
      -o res
```

3.2.3 Capturing and Filtering Raw Results

Mercury BLASTN exhibits sensitivity to significant alignments that is typically comparable to that of NCBI BLASTN. Tests with an E-value threshold of 10^{-5} indicate that it typically returns between 98 and 100% of the alignments that NCBI BLASTN would return, as well as some other significant alignments that NCBI BLASTN misses. However, it is possible to increase sensitivity even further, at some cost to computational efficiency, by running Mercury BLASTN with a more permissive E-value threshold (say, 1.0) and then capturing only the most significant results.

To support this usage, it is possible to capture the raw, unpostprocessed results of an `HgBlastall` search and postprocess them manually. Adding the –preserverun option to the `HgBlastall` command line and providing a *nonexistent* directory name <dirname> as its argument will cause the raw results to be saved in directory <dirname>. These results may be filtered later using the `HgMFAPost` command.

If <dirname> is not an absolute path, it is assumed to be a subdirectory of `/s0/scratch/rn`.

Note that in order to use `HgMFAPost`, you must save *both* the preprocessed query files and the raw results for a search. As above, it is strongly recommended that all these files be saved to the local RAID for performance reasons.

Example

The following example captures the raw output from a comparison of the query query.fna to the database db with an E-value threshold of 1.0, then extracts as final output only those alignments with E-value at most 10^{-5}. The resulting set of alignments is typically larger than that obtained by running the original search with E-value 10^{-5}.

```
HgBlastall –p blastn –i query.fna –d db –e 1.0 –o res1.0
      –preserveqy savedqy –preserverun results1.0

HgMFAPost –mfafile /s0/scratch/qy/savedqy –resfile /s0/scratch/rn/results1.0
      –sumfile res2 –e 1e-5
```

3.2.3.1 The HgMFAPost Command

Syntax

```
HgMFAPost –mfafile <dirname> –resfile <dirname>
      –sumfile <filename> [–e <evalue>]
```

-mfafile	directory containing a preprocessed query
-resfile	directory containing unpostprocessed results
-sumfile	file to hold preprocessed output
-e	maximum E-value threshold for including results in output (real-valued; default is to include all results)

`HgMFAPost` takes three required arguments: a directory containing a preprocessed query (-mfafile), a directory containing raw, un-postprocessed results for that query (-resfile), and the name of a file to hold the final output after postprocessing (-sumfile). It postprocesses the results directory into a single output file, rewriting sequence coordinates to undo any splitting performed during query preparation and sorting the results.

The query and raw results directories may be obtained from a search by running `HgBlastall` with the –preserveqy and –preserverun arguments, respectively. The query directory may alternatively be generated manually by running `HgPreparequery`.

The –e argument may be used to emit only a subset of alignments from the raw results. In particular, specifying '-e <evalue>' emits only alignments with E-value at most <evalue>.