libsdbf

Similarity Digest Bloom Filter Library

sdhash version 3.3

Vassil Roussev, Candice Quates

http://sdhash.org

July 15, 2013

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 bloom_filter Class Reference

bloom_filter class

`#include <bloom_filter.h>`

Public Member Functions

- bloom_filter (uint64_t size, uint16_t hash_count, uint64_t max_elem, double max_fp)

    base constructor
- bloom_filter (string indexfilename)

    construct from file - not add to master or fold up.
- bloom_filter (uint8_t ∗data, uint64_t size, int id, int bf_elem_ct, uint16_t hamming)

    construct bloom filter from buffer
- ∼bloom_filter ()

    destructor
- bool insert_sha1 (uint32_t ∗sha1)

    insert SHA1 hash
- bool query_sha1 (uint32_t ∗sha1)

    query SHA1 hash
- uint64_t elem_count ()

    return element count
- double est_fp_rate ()

    return estimate of false positive rate
- double bits_per_elem ()

    return bits per element
- string name () const

    name associated with bloom filter
- void set_name (string name)

    change name associated with bloom filter
- void fold (uint32_t times)

    fold a large bloom filter onto itself
- int add (bloom_filter ∗other)

    add another same-sized bloom filter to this one
- int write_out (string filename)

    write bloom filter to .idx file
- int bloom_id ()

    id associated with bloom filter (used for grouping)

2.1.1 Detailed Description

bloom_filter: a Bloom filter class.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 bloom_filter::bloom_filter (uint64_t *size*, uint16_t *hash_count*, uint64_t *max_elem*, double *max_fp*)

Create new empty bloom filter

Parameters

size	of bloom filter
hash_count	number of hashes for each insertion or query
max_elem	max element size (0 ok)
max_fp	max false positive rate (0 ok)

2.1.2.2 bloom_filter::bloom_filter (string *indexfilename*)

Read bloom filter from a file

Parameters

indexfilename	file to read

2.1.2.3 bloom_filter::bloom_filter (uint8_t * *data*, uint64_t *size*, int *id*, int *bf_elem_ct*, uint16_t *hamming*)

Creates bloom filter from existing buffer of bloom filter data. Experimental: sized for sdbf 256-byte bloom filters at the moment

Parameters

data	buffer of bloom filter data
size	of bloom filter data
id	identifier for clustering bloom filters
bf_elem_ct	# of elements in filter
hamming	weight of filter

2.1.2.4 bloom_filter::∼bloom_filter ( )

Destroys bloom filter and frees buffer

2.1.3 Member Function Documentation

2.1.3.1 int bloom_filter::add (bloom_filter * *other*)

Adds another bloom filter to this one

Parameters

other	bloom filter

Returns

0 if successful 1 if not the same size

2.1.3.2 double **bloom_filter::bits_per_elem** ( )

Returns bits per element in bloom filter

Returns

estimate

2.1.3.3 uint64_t **bloom_filter::elem_count** ( )

Returns number of elements present in bloom filter

Returns

number of elements

2.1.3.4 double **bloom_filter::est_fp_rate** ( )

Returns estimated false positive rate (not implemented)

Returns

estimate

2.1.3.5 void **bloom_filter::fold** (uint32_t *times*)

Folds bloom filter by half N times by or'ing the second half of the bloom filter onto the first half.

Parameters

times	amount of times to fold filter

2.1.3.6 bool **bloom_filter::insert_sha1** (uint32_t ∗ *sha1*)

Inserts hash data into this bloom filter

Parameters

sha1	buffer of sha1 hash values

Returns

exists or not exists

2.1.3.7 string **bloom_filter::name** ( ) const

Returns name associated with bloom filter

Returns

name

2.1.3.8 bool **bloom_filter::query_sha1** (uint32_t * *sha1*)

Queries this bloom filter with hash data

Parameters

sha1	buffer of sha1 hash values

Returns

exists or not exists

2.1.3.9 void **bloom_filter::set_name** (string *name*)

Changes name associated with bloom filter

Parameters

name	new name

2.1.3.10 int32_t **bloom_filter::write_out** (string *filename*)

Writes bloom filter out to a file.

Parameters

filename	file to be written

Returns

status -1 if compression fails, -2 if cannot open file

The documentation for this class was generated from the following files:

- sdbf/bloom_filter.h
- sdbf/bloom_filter.cc

2.2 sdbf Class Reference

sdbf class

`#include <sdbf_class.h>`

Public Member Functions

- sdbf (FILE *in)

  to read formatted sdbfs from open file pointer
- sdbf (const char *filename, uint32_t dd_block_size)

  to create new from a single file

- sdbf (const char ∗name, std::istream ∗ifs, uint32_t dd_block_size, uint64_t msize, index_info ∗info)

  to create by reading from an open stream
- sdbf (const char ∗name, char ∗str, uint32_t dd_block_size, uint64_t length, index_info ∗info)

  to create from a c-string
- ∼sdbf ()

  destructor
- const char ∗ name ()

  object name
- uint64_t size ()

  object size
- uint64_t input_size ()

  source object size
- int32_t compare (sdbf ∗other, uint32_t sample)

  matching algorithm, take other object and run match
- string to_string () const

  return a string representation of this sdbf
- string get_index_results () const

  return results of index search
- uint8_t ∗ clone_filter (uint32_t position)

  return a copy of an individual bloom filter from this sdbf

Static Public Attributes

- static class sdbf_conf ∗ config = new sdbf_conf(1, FLAG_OFF, _MAX_ELEM_COUNT, _MAX_ELEM_COU-NT_DD)

  global configuration object

Friends

- std::ostream & operator<< (std::ostream &os, const sdbf &s)

  output operator
- std::ostream & operator<< (std::ostream &os, const sdbf ∗s)

  output operator

2.2.1 Detailed Description

sdbf: a Similarity Digest Bloom Filter class.

Examples:

sdbf_test.cc.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 sdbf::sdbf (FILE ∗ *in*)

Reads an already generated sdbf from open file. Throws exceptions in case of bad formatting.

Parameters

in	FILE∗ open formatted as list of sdbfs

2.2.2.2 sdbf::sdbf (const char ∗ *filename,* uint32_t *dd_block_size*)

Create new sdbf from file. dd_block_size turns on "block" mode.

Parameters

filename	file to hash
dd_block_size	size of block to process file with. 0 is off.

2.2.2.3 sdbf::sdbf (const char ∗ *name,* std::istream ∗ *ifs,* uint32_t *dd_block_size,* uint64_t *msize,* index_info ∗ *info*)

Generates a new sdbf, with a maximum size read from an open stream. dd_block_size enables block mode.

Parameters

name	name of stream
ifs	open istream to read raw data from
dd_block_size	size of block to divide data with. 0 is off.
msize	amount of data to read and process
info	block of information about indexes

2.2.2.4 sdbf::sdbf (const char ∗ *name,* char ∗ *str,* uint32_t *dd_block_size,* uint64_t *length,* index_info ∗ *info*)

Generates a new sdbf, from a char ∗string dd_block_size enables block mode.

Parameters

name	name of stream
str	input to be hashed
dd_block_size	size of block to divide data with. 0 is off.
length	length of str to be hashed
info	block of information about indexes

2.2.2.5 sdbf::∼sdbf ( )

Destroys this sdbf

2.2.3 Member Function Documentation

2.2.3.1 uint8_t ∗ sdbf::clone_filter (uint32_t *position*)

Clones a copy of a single bloom filter in this sdbf.

Warning: 256-bytes long, not terminated, may contain nulls.

Parameters

position	index of bloom filter

Returns

uint8_t∗ pointer to 256-byte long bloom filter

2.2.3.2 int32_t sdbf::compare (sdbf ∗ *other*, uint32_t *sample*)

Compares this sdbf to other passed sdbf, returns a confidence score

Parameters

other	sdbf∗ to compare to self
sample	sets the number of BFs to sample - 0 uses all

Returns

int32_t confidence score

Examples:

sdbf_test.cc.

2.2.3.3 uint64_t sdbf::input_size ( )

Returns the size of the data that the hash was generated from.

Returns

uint64_t length value

2.2.3.4 const char ∗ sdbf::name ( )

Returns the name of the file or data this sdbf represents.

Returns

char∗ of file name

2.2.3.5 uint64_t sdbf::size ( )

Returns the size of the hash data for this sdbf

Returns

uint64_t length value

2.2.3.6 string sdbf::to_string ( ) const

Encode this sdbf and return it as a string.

Returns

std::string containing sdbf suitable for display or writing to file

The documentation for this class was generated from the following files:

- sdbf/sdbf_class.h
- sdbf/sdbf_class.cc
- sdbf/sdbf_core.cc

2.3 sdbf_conf Class Reference

```
#include <sdbf_conf.h>
```

Public Member Functions

- sdbf_conf (uint32_t thread_cnt, uint32_t warnings, uint32_t max_elem_ct, uint32_t max_elem_ct_dd)

  constructor: set defaults
- ∼sdbf_conf ()

  destructor

Public Attributes

- uint32_t thread_cnt

  number of pthreads available
- uint32_t max_elem

  maximum elements per bf
- uint32_t max_elem_dd

  maximum elements per bf - dd mode
- uint32_t warnings

  whether to process warnings

2.3.1 Detailed Description

Configuration object for sdbf classes. Used as a static class member to provide globally tunable defaults and access to bit counting structures.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 sdbf_conf::sdbf_conf (uint32_t *thread_cnt,* uint32_t *warnings,* uint32_t *max_elem_ct,* uint32_t *max_elem_ct_dd*)

constructor for sdbf_conf object. takes thread_count, warnings, max elements in BF

The documentation for this class was generated from the following files:

- sdbf/sdbf_conf.h
- sdbf/entr64.cc
- sdbf/sdbf_conf.cc

2.4 sdbf_set Class Reference

sdbf_set class

```
#include <sdbf_set.h>
```

Public Member Functions

- sdbf_set ()

  creates blank sdbf_set
- sdbf_set (bloom_filter ∗index)

creates blank sdbf_set with index
- sdbf_set (const char ∗fname)

  loads an sdbf_set from a file
- ∼sdbf_set ()

  destructor
- class sdbf ∗ at (uint32_t pos)

  accessor method for individual hashes
- void add (class sdbf ∗hash)

  adds a single hash to this set
- void add (sdbf_set ∗hashset)

  adds the items in another set to this set
- uint64_t size ()

  Returns the number of sdbfs in this set.
- uint64_t input_size ()

  Computes the data size of this set.
- uint64_t filter_count ()
- void compare_all (int32_t threshold)

  Compares all objects in a set to each other.
- std::string compare_all_quiet (int32_t threshold, int32_t thread_count)
- void compare_to (sdbf_set ∗other, int32_t threshold, uint32_t sample_size)

  queries one set for the contents of another
- std::string compare_to_quiet (sdbf_set ∗other, int32_t threshold, uint32_t sample_size, int32_t thread_count)
- std::string to_string () const

  return a string which contains the output-encoded sdbfs in this set
- std::string index_results () const

  return a string which contains the results of this set's index seraching
- int empty ()

  is this empty?
- std::string name () const

  retrieve name of set
- void set_name (std::string name)

  name this set.
- void set_separator (char sep)

  change output separator
- void vector_init ()

  setup bloom filter vector

Public Attributes

- class bloom_filter ∗ index

  index for this set
- std::vector< class
  bloom_filter ∗ > ∗ bf_vector

  giant bloom filter vector for this set

Friends

- std::ostream & operator<< (std::ostream &os, const sdbf_set &s)

  output operator
- std::ostream & operator<< (std::ostream &os, const sdbf_set ∗s)

  output operator

2.4.1 Constructor & Destructor Documentation

2.4.1.1 sdbf_set::sdbf_set ( )

Creates empty sdbf_set

2.4.1.2 sdbf_set::sdbf_set (bloom_filter ∗ *index*)

Creates empty sdbf_set with an index

Parameters

index	to insert new items into

2.4.1.3 sdbf_set::sdbf_set (const char ∗ *fname*)

Loads all sdbfs from a file into a new set

Parameters

fname	name of sdbf file

2.4.2 Member Function Documentation

2.4.2.1 void sdbf_set::add (class sdbf ∗ *hash*)

Adds a single hash to this set

Parameters

hash	an existing sdbf hash

2.4.2.2 void sdbf_set::add (sdbf_set ∗ *hashset*)

Adds all items in another set to this set

Parameters

hashset	sdbf_set∗ to be added

2.4.2.3 class sdbf ∗ sdbf_set::at (uint32_t *pos*)

Accessor method for a single sdbf∗ in this set

Parameters

pos	position 0 to size()

Returns

sdbf∗ or NULL if position not valid

2.4.2.4 void **sdbf_set::compare_all** (int32_t *threshold*)

Compares each sdbf object in target to every other sdbf object in target and prints the results to stdout

Parameters

threshold	output threshold, defaults to 1

2.4.2.5 std::string **sdbf_set::compare_all_quiet** (int32_t *threshold,* int32_t *thread_count*)

Compares each sdbf object in target to every other sdbf object in target and returns the results as a list stored in a string

Parameters

threshold	output threshold, defaults to 1
thread_count	processor threads to use, 0 for all available

Returns

std::string result listing

2.4.2.6 std::string **sdbf_set::compare_to_quiet** (**sdbf_set** ∗ *other,* int32_t *threshold,* uint32_t *sample_size,* int32_t *thread_count*)

Compares each sdbf object in other to each object in this set, and returns the results as a list stored in a string.

Parameters

other	set to compare to
threshold	output threshold, defaults to 1
sample_size	size of bloom filter sample. send 0 for no sampling
thread_count	processor threads to use, 0 for all available

Returns

std::string result listing

2.4.2.7 int **sdbf_set::empty** ( )

Checks empty status of container

Returns

int 1 if empty, 0 if non-empty

2.4.2.8 uint64_t **sdbf_set::filter_count** ( )

Returns the size of the set's own bloom_filter vector.

2.4.2.9 std::string **sdbf_set::index_results** ( ) const

Generates a string representing the indexing results of this set

2.4.2.10 uint64_t **sdbf_set::input_size** ( )

Computes the data size of this set, from the input_size() values of its' content sdbf hashes.

Returns

uint64_t total of input sizes

2.4.2.11 std::string **sdbf_set::name** ( ) const

Retrieve name of this set

Returns

string name

2.4.2.12 void **sdbf_set::set_name** (std::string *name*)

Change name of this set

Parameters

name	of string

2.4.2.13 void **sdbf_set::set_separator** (char *sep*)

Change comparison output separator

Parameters

sep	charactor separator for output

2.4.2.14 uint64_t **sdbf_set::size** ( )

Number of items in this set

Returns

uint64_t number of items in this set

2.4.2.15 std::string **sdbf_set::to_string** ( ) const

Generates a string which contains the output-encoded sdbfs in this set

Returns

std::string containing sdbfs.

2.4.2.16 void **sdbf_set::vector_init** ( )

Sets up bloom filter vector. Should also be called by server process when done hashing to a set

The documentation for this class was generated from the following files:

- sdbf/sdbf_set.h
- sdbf/sdbf_set.cc

Chapter 3

Example Documentation

3.1 sdbf_test.cc

A very short example program using sdbf.

```
/* sdbf_test.cc shortest possible test program for sdbf
   author: candice quates

g++ sdbf_test.cc -o sdbf_test ../libsdbf.a -lcrypto -lc -lm -lpthread

*/


#include <iostream>
#include <stdint.h>

#include "../sdbf/sdbf_class.h"
#include "../sdbf/sdbf_defines.h"

using namespace std;

int
main() {
    uint32_t res1;
    /// create new sdbf from binary of built sdhash file, no parallelism
    sdbf *test1 = new sdbf("../sdhash", 0);
    /// create new sdbf from binary of Doxygen file, 16KB to a bloom filter
    sdbf *test2 = new sdbf("../Doxyfile", 16*1024);

    /// display our hashes
    cout << test1 ;
    cout << test2 ;
    /// Compare test 1 with test2, and print the resulting score.
    res1=test1->compare(test2,0,0);

    uint8_t* stuff;
    cout << "test1 vs test2 " << res1 << "\n";
    return 0;
}
```

Index