About NEST

1. About NEST

NEST supports constant-scale hashing scheme with two salient features, i.e., locality awareness and flat addressing. The design details are shown in our INFOCOM 2013 paper:

Yu Hua, Bin Xiao, Xue Liu, "NEST: Locality-aware Approximate Query Service for Cloud Computing", Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM), April 2013, pages: 1327-1335.

2. NEST Components

The components of source codes consist of three parts, including the LSH, cuckoo hashing and their collaborative implementations.

LSH part:

lsh.h: the header file of LSH
lsh.c: the LSH implementation
lsh.conf: the configurations of LSH parameters
lsh_test.c: the test codes and the details in the file ”test.README”

Cuckoo Hashing part：

cuckoo.h: the header file of cuckoo hashing
cuckoo.c: the cuckoo hashing implementations
cuckoo_test.c: the test codes and the details in the file test.README

NEST part：

nest.h: the header file of NEST
nest.c: the implementation of NEST
nest.conf: the configurations of NEST
nest_test.c: the test codes and the details in the file “test.README” (the fourth part)

Other files：

string_generator.c: Generate random strings for evaluate the cuckoo hashing
vectors_generator.c: Generate multi-dimensional random vectors as described in the file “trace.README”

3. The interface in NEST

The interfaces support the construction, release functions in NEST, as well as ANN query and status query. To offer the flexibility, NEST offers the self-defined interfaces to allow users to define hash functions, as well as the “kick-out” methodology.

NEST Construction：

int initNestParam(const char *config_file)：

Initialize NEST parameters that are used in the entire NEST execution. The parameter configurations are shown in “nest.conf”.

HashTable **hashTablesCreate(void (*free_data)(void *ptr), void (*free_info)(void *ptr))：
This interface is used to create the hash tables, which is the basic structure of NEST storage and query modules. Users need to provide the function for releasing memory of inserted data. This includes memory release function and auxiliary functions.

Nest *nestCreate(HashTable **hash_tables)：
Create Nest and use the parameters from the hash table “hashTablesCreate” as the input

NEST Operations：
Nest *nestInsertItem(Nest *nest, void *data, void *info)：
This interface is used to insert data. The input for insertion includes the data and optional auxiliary information.

Nest *nestRemoveItem(Nest *nest, void *data, int (*match)(void *data, void *ptr)):
This interface is used to remove specific data by the “match” function to locate the data to be removed.

NNResult *nestGetNN(Nest *nest, void *data)：
This interface is used to query the nearest data. The results are returned by the structure “NNResult”, which includes the number of results and lists.

void *nestGetItem(Nest *nest, void *data, int (*match)(void *data, void *ptr)):
This interface is used to query a single data item in NEST by “match” function to locate the matching items.

NestSetFindPosMethod(nest, m)
This function supports self-defined “kick-out” macro definition in cuckoo hashing, in which “nest” is the pointer of NEST，m is the “kick-out” function. The function is defined as “int findOptPos(Nest *nest, HashValue *hash_value, void *data)”;

NestSetHashMethods(nest, m)
This function supports the self-defined hash functions, in which “nest” is the pointer in NEST, “m” is the pointer array of hash functions. The function is defined as “unsigned long hashfunc (void *data, void *other)”. Note that the number of hash function pointers is the same as that of hash tables.

void nestReport(Nest *nest)：
This function is to print the status of NEST structure, including the size of hash table, the number of item insertion, overhead and insertion latency.

NEST release：
void freeNestParam()：
Release the configurations of NEST

void nestDestroy(Nest *nest)：
Release NEST structure

void freeNNResult(NNResult *res)：
Release the ANN query results

4. An example:
“lsh_test” is used to evaluate the LSH performance and the test dataset is generated by “vectors_generator”；

“cuckoo_test” is used to evaluate cuckoo hashing performance and the test dataset is generated by ”string_generator”；

“nest_test” is used to evaluate NEST and the test dataset is generated by “vectors_generator”

The details are shown in “test.README” and “trace.README”.

The evaluation platform is configured by gcc (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1 and Linux version 3.0.34 32-bit systems.