TinyTable in C++

PROJECT REPORT

Yulia Sher | Project in Parallel & Distributed Programming | Spring 2017

Overview

TinyTable is a compact hash table that can be used to efficiently maintain networking flow statistics. Its functionality is very important for networking devices, but can also be used as a general purpose highly space efficient.

TinyTable hash based algorithm that supports membership queries, removals and multiplicity queries documented in the following report:

http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi/2015/CS/CS-2015-04

This report describes a TinyTable C++ implementation.

TinyTable Implementation

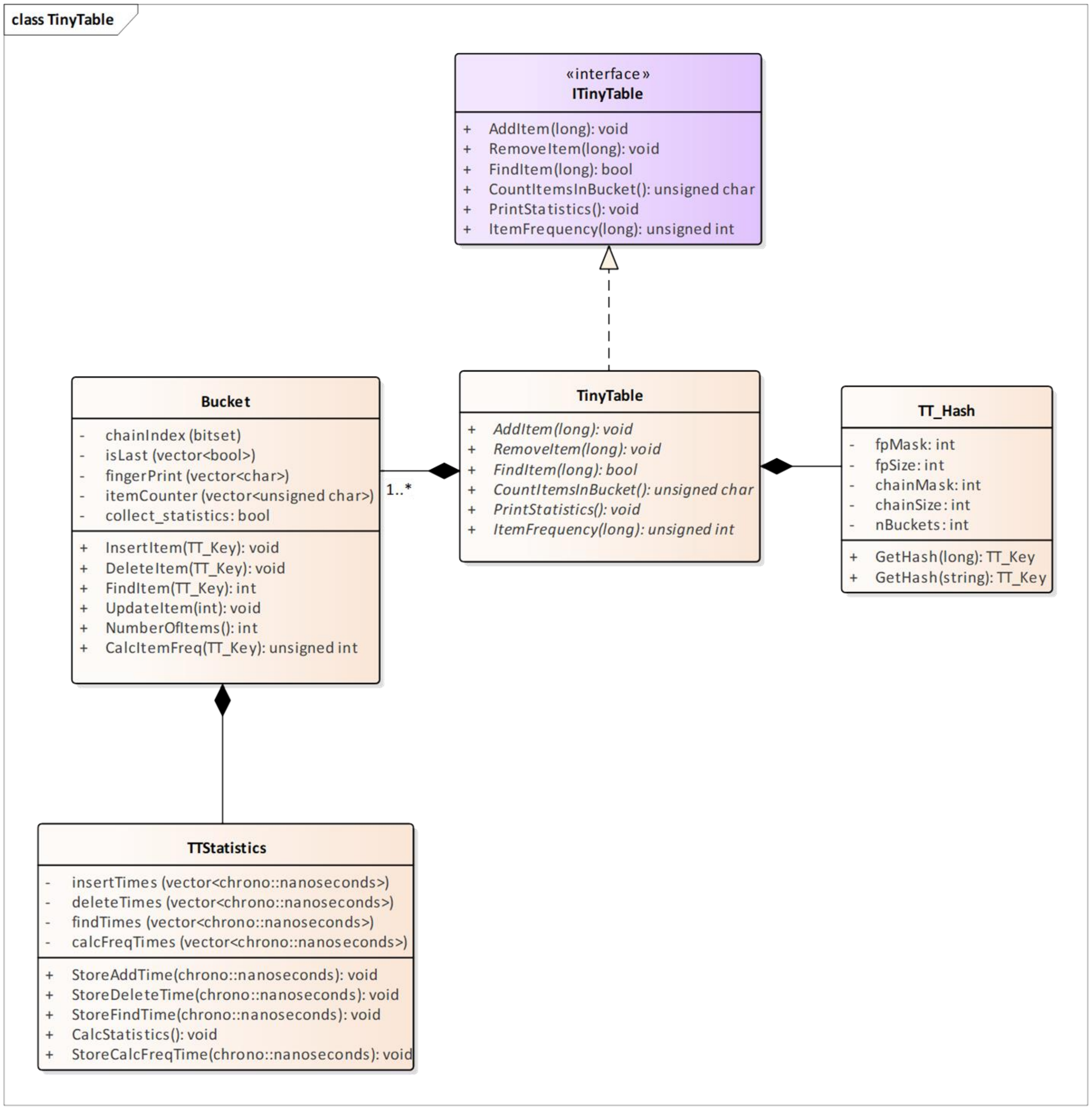
FUNCTIONALITY

The following operations were implemented to support TinyTable data structure:

- Add Operation – allows to add an item to the table (an item can be both a string and a long).
- Remove Operation – allows to remove an item from the table.
- Query Operation – allows to determine whether a table contains an item.
- Multiplicity Query Operation - allows to determine the number of times an element appears in the table.
- Counting Items in Bucket Operation – allows to determine the number of items in the bucket.
- Print Statistics Operation – allows to visualize runtime statistics that were collected during TinyTable execution.

ARCHITECTURE & DESIGN

Class Model



class TinyTable
«interface»
ITinyTable
+ AddItem(long): void
+ RemoveItem(long): void
+ FindItem(long): bool
+ CountItemsInBucket(): unsigned char
+ PrintStatistics(): void
+ ItemFrequency(long): unsigned int
Bucket
- chainIndex (bitset)
- isLast (vector<bool>)
- fingerPrint (vector<char>)
- itemCounter (vector<unsigned char>)
- collect_statistics: bool
+ InsertItem(TT_Key): void
+ DeleteItem(TT_Key): void
+ FindItem(TT_Key): int
+ UpdateItem(int): void
+ NumberOfItems(): int
+ CalcItemFreq(TT_Key): unsigned int
1..*
TinyTable
+ AddItem(long): void
+ RemoveItem(long): void
+ FindItem(long): bool
+ CountItemsInBucket(): unsigned char
+ PrintStatistics(): void
+ ItemFrequency(long): unsigned int
TT_Hash
- fpMask: int
- fpSize: int
- chainMask: int
- chainSize: int
- nBuckets: int
+ GetHash(long): TT_Key
+ GetHash(string): TT_Key
TTStatistics
- insertTimes (vector<chrono::nanoseconds>)
- deleteTimes (vector<chrono::nanoseconds>)
- findTimes (vector<chrono::nanoseconds>)
- calcFreqTimes (vector<chrono::nanoseconds>)
+ StoreAddTime(chrono::nanoseconds): void
+ StoreDeleteTime(chrono::nanoseconds): void
+ StoreFindTime(chrono::nanoseconds): void
+ CalcStatistics(): void
+ StoreCalcFreqTime(chrono::nanoseconds): void

ITinyTable – an interface for the TinyTable operations:

- AddItem – adds an item to the table. In case an item already exists in the table, its counter is updated. An item to be added to the table can be both of type long and std::string.
- RemoveItem – removes an item from the table. In case an item appearance equal to one it is removed from the table, otherwise its counter is updated.
- FindItem – checks if an item appears in the table. Returns true or false.
- ItemFrequency – returns how many times an item appears in the table.
- CountItemsInBucket – returns number of items in a specific bucket.
- PrintStatistics – prints performance statistics of the TinyTable operations.

TinyTable – a main data structure of the table. Implements an ITinyTable interface. Contains a vector of buckets – one or more (defined by number of elements to be added to the table) and a hash function(TT_Hash). When created with use_statistics flag = true performs statistics collection.

Bucket – represents a single bucket. Composed of:

- chainIndex (a 64-bit bitset) – manages bit per chain
- isLast (a vector of Booleans) – holds true or false for each fingerprint added to the table
- fingerPrint (a vector of chars) – holds fingerprints of elements added to the table
- itemCounter (a vector of unsigned chars) – holds a counter for each element added to the table

Bucket receives an element as a key(TT_Key) and performs the following operations:

- InsertItem – adds a new element to the table
- FindItem – checks whether an element appears in the table
- UpdateItem - updates a counter of the element that was already added to the table
- DeleteItem – deletes an element from the table or updates a counter in case there is more than one element left in the table.
- NumberOfItems – returns the number of elements in the bucket
- CalcItemFreq – returns the number of times an element appears in the table

TT_Hash – represents a hash function used to provide a key(TT_Key) for each element to be operated by the table.

Composed of:

- nBuckets – number of buckets in the table
- fpSize – a fingerprint size (in bits)
- chainSize – a chain size (in bits)

- fpMask – a mask used to produce a fingerprint attribute of the key (defined by fpSize)
- chainMask – a mask used to produce a chain attribute of the key (defined by chainSize)

GetHash operations applies a hash function on the element plus additional shifts to prevent overflows and returns a key(TT_Key).

TT_Key – represents a key value to be stored in the table.

Composed of:

- bucketId (an unsigned char)
- chainId (an unsigned char)
- fingerprint (a char)

PROGRAMMING

C++

A Modern C++ (C++ 11/14/17) was used for the TinyTable implementation.

Standard containers such as vector and bitset were used with STL algorithms for better runtime results. Hash function used in TT_Hash is also a STL implementation.

Chrono STL library was used to measure execution time intervals.

Environment

The project was developed in VS2017 and compiled with V141 toolset.

Results

The tests were performed on Intel® Core™ i5-7200U CPU @ 2.50GHz 2.70 GHz, 8.0GB(RAM), Windows 10 64-bit Operating System

The trace file equinix-chicago.dirA.20151217-130000-130500.trace_part_1 contains 2M elements was used to perform Add, Find, Find Frequency and Delete operations.

The test was repeated for 10 times and the following runtime statistics were collected:

- Add operation – 458 ns
- Find operation – 204 ns
- Find Frequency operation – 119 ns
- Delete operation – 218 ns

Additional tests were performed to validate the correctness of TinyTable operations.

How to use the application

Four arguments should be provided as program arguments:

- File containing the elements to be added to the table
- File containing the elements to be found
- File containing the elements to be checked for frequency
- File containing the elements to be deleted

In case one of the above is not required an empty string should be passed as an argument.