The Hashing Framework

Finding things fast.

applied informatics

Overview

- Motivation
- Implementations
- Performance

Motivation

- std::map and std::set give a performance guarantee of O(log n)
- collections using hashing can show a better real-world performance than their STL counter parts

Implementations

- LinearHashTable
- HashSet
- HashMap
- HashTable (deprecated)
- SimpleHashTable (deprecated)

LinearHashTable

- implements the basic data structure used by HashMap and HashSet
  #include “Poco/LinearHashTable.h”
- user must provide a hash function
  Poco/Hash.h contains predefined ones for integral numbers and std::string
  Poco::LinearHashTable<Key, Hash = Poco::Hash<Key>
- performs linear hashing, no performance deterioration with inserts/deletes (no rehashing needed when out of data)

HashMap

- std::map like functionality
  #include “Poco/HashMap.h”
- use like a map
  same interface, even iterators are there

HashSet

- std::set like functionality
  #include “Poco/HashSet.h”
- use like a set
  same interface, even iterators are there

Deprecated Classes

- HashTable, SimpleHashTable
  - uses Poco::HashFunction !
  - no STL like interface
  - no iterator
  - simpler but faster

SimpleHashTable (deprecated)

- the fastest implementation
  #include “Poco/SimpleHashTable.h”
- limitations
  - no remove
  - static fixed size
    - when inserting into a full table: exception
  - simple overflow handling: scan for next free hole
  - wastes memory: capacity > elemCount

HashTable (deprecated)

- #include “Poco/HashTable.h”
- uses overflow maps to handle collisions
  - when created with a size of 1 it is a map
- supports remove operations

Recommendations

- use HashMap/HashSet where possible
- don't use HashTable at all
  - it is only slightly faster than HashMap
- when your application depends on every single CPU cycle and you need map functionality, then and only then, use SimpleHashTable
  - approx 30 % faster than HashMap

Performance

- always depends on the usage scenario and the size of the data
- hashing is approximately two times faster than the STL containers
  - SimpleHashTable adds another 30 % when configured properly.
- the larger the collection the higher are the performance gains

applied informatics

www.appinf.com | info@appinf.com
T +43 4253 32596 | F +43 4253 32096