Bivas Maiti
bmaiti@iu.edu

Key Value Store Implementation

A key value storage mechanism is implemented, which is the ground word for cache utilization in a lot of systems. The functions implemented are Get Key, Set Key, Delete Key, Reset Cache, and Initialize Cache. Accounting functions implemented are get cache info and most popular keys. This uses Least Frequently Used(LFU) as a Cache Replacement Policy. To run the test case, run the following command in xinu:

xsh $ cache_test

Functions

Set Key(key, value)

The algorithm to set a key is as follows:
Check if key exists.
If it does, replace the old value with the new value and record the hit on the key.
If key does not exist, try to find an empty place in the cache.
    If found, insert the key value pair into the place.
If not, get the least frequently hit item from the cache. Replace the key and value of the item with the new key and value

Get Key(key)

Iterate through the cache to check if key exists.
IF key exists return the corresponding value.
If key does not exist, return null.

Delete Key(key)

Iterate through the cache to check if key exists.
IF key exists remove the key from the cache making the place empty and return TRUE
If key does not exist, return FALSE

Reset Cache

Iterate through the cache
If key exists at a place, remove the key and corresponding value.

Get Cache Info(kind)

Check the kind of information the caller function wants. Send the information from the global accounting variables. Global accounting variables are: total_hits, total_accesses, total_set_success, cache_size, num_keys total_evictions. These get changed every time the get/set/delete/reset functions are called.

Most Popular Keys(k)

Sort the cache according to the most hits occurred in an item. Return first k keys of the items.