Functional Specification

for An Efficient Wait-free

Resizable Hash Table project

Tal Gelbard – X – [talgelbard@campus.technion.ac.il](mailto:talgelbard@campus.technion.ac.il)

Alon Libling – 307886192 – [lalon@campus.technion.ac.il](mailto:lalon@campus.technion.ac.il)

# overview

A data structure of efficient wait-free resizable parallel hash table, based on [this](https://tropars.github.io/downloads/pdf/publications/spaa2018-FKR-WF_ext_hashing.pdf) paper using buckets and PSim in order to implement a dictionary of <key, value> pairs in CPP language.

להרחיב

# Main functional entities

The functional entities are CPP programmers, using the following API:

* *Insert* (also updates the value of an existing key).
* *Look-up*.
* *Delete*.

# Must have features

* Wait-free: look-up operations will be allowed without any synchronization since the look-up is the most common action.
* Resizable: the data structure size will be dynamic in accordance with the input size.
  + Update operations (insert and delete) on different buckets will be executed in parallel when no resizing actions being handled.

# Nice to have features

Operations such as:

* Visual printing of the data structure using the buckets.
* Get all keys/values, get size. Return all keys that comply a certain condition.
* Convert to set (using a tree).

# Out of scope features/issues

* Special operations that are been executed on all items:
  + Receives *std∷function* or *generic lambda* and activates it on all the items.
* Shrink table size by eliminating the least requested keys operation.
* Nested buckets in order to maintain a smaller DState objects.