universite PARIS-SACLAY | IUT D'ORSAY

Tarides

CacheCache, an efficient cache library

23/06/2022

Charlène Gros

CacheCache, an efficient cache library

- Project context
    - Tarides
    - Irmin
    - CacheCache
- Missions
    - Support two strategies of use
    - Implementation of LFU strategy
    - Tests, Benchmarks and Formal Specifications
- Thanks

Context

Tarides

- Created in 2018
- Work in OCaml
- Open-source
- International collaboration
- Weekly meeting

Irmin

- Git like database
- Versioned key-value stores
- Branchable and mergeable
- Customizable
- Critical application : need efficiency

CacheCache

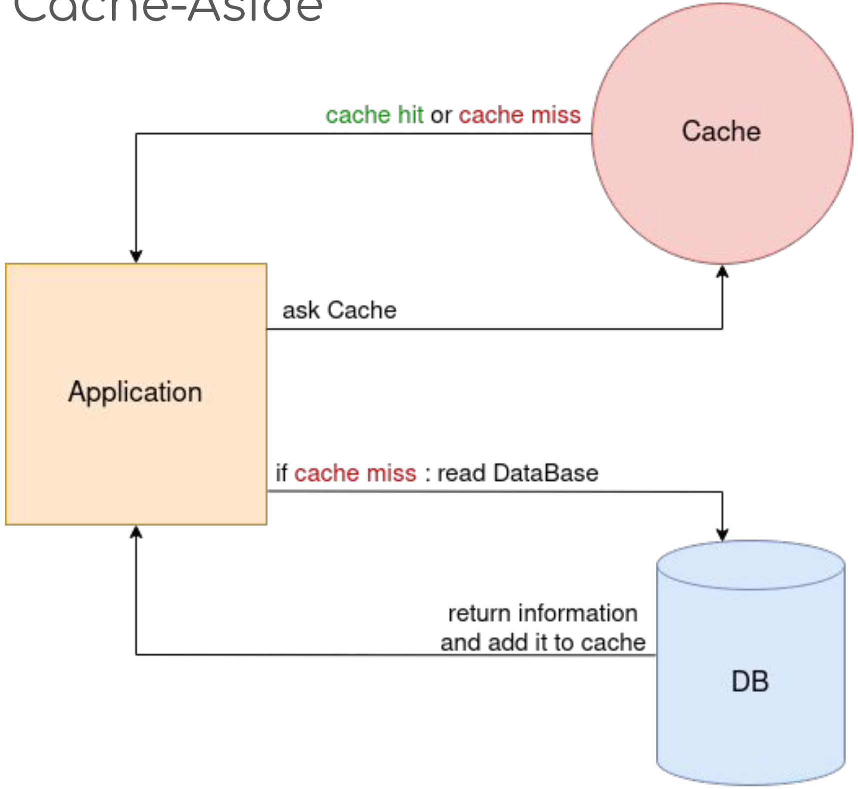
- Local source faster than external one
- Exists various Cache replacement policies
- In OCaml
- Optimizing algorithm (Constant time complexity)
- Library designed for Irmin
- To let Irmin be more efficient

Missions

Usage strategies

Cache usage strategies

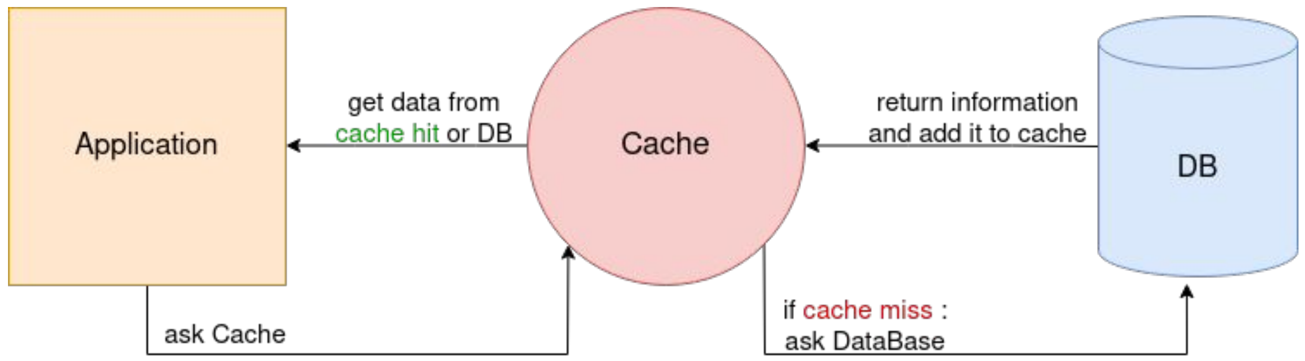
Cache-Aside



```
try
   Cache.find cache key
with Not_found ->
   DB.find source key
```

Cache usage strategies

Cache-Through



Cache usage strategies

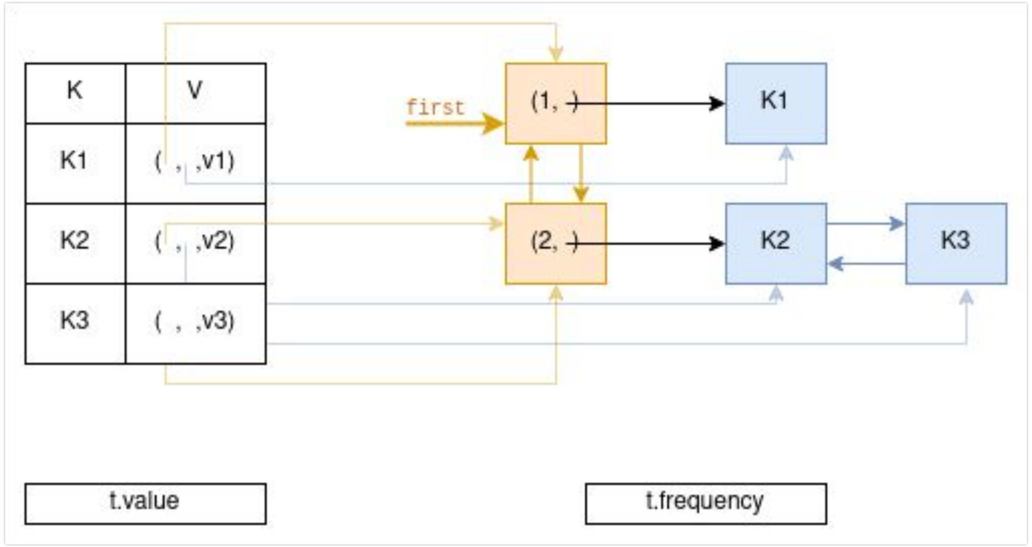
Cache-Through

```
module Make (Cache : Cache) (DB with type key = Cache.key) :
sig
 include DB

 val create : int -> t

 module Cache : sig
   (*...*)
   val clear : t -> unit
   val mem : t -> key -> bool
   val find : t -> key -> value
   val remove : t -> key -> unit
 end
end
```

Least frequently used (LFU)

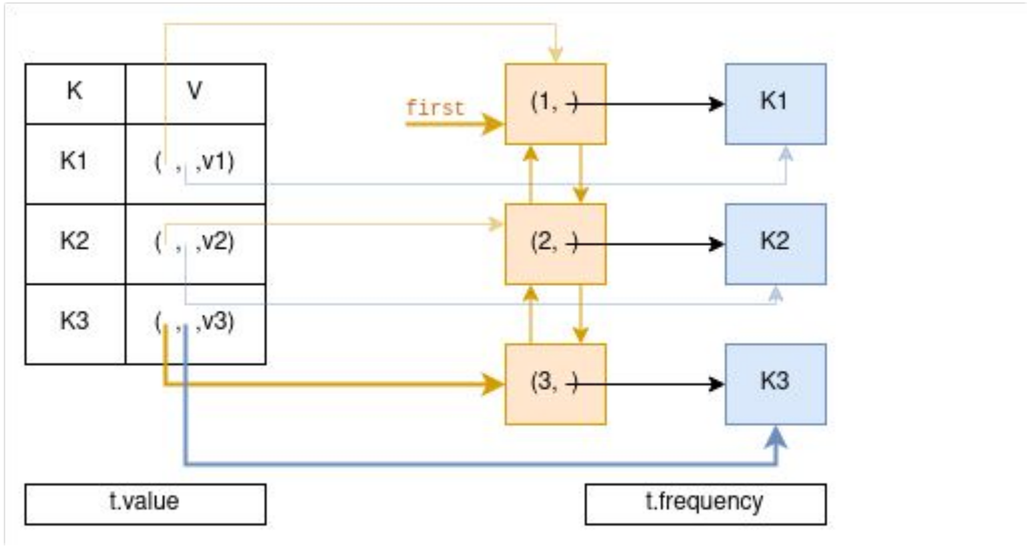
LFU Data-structure

Cache size = 3



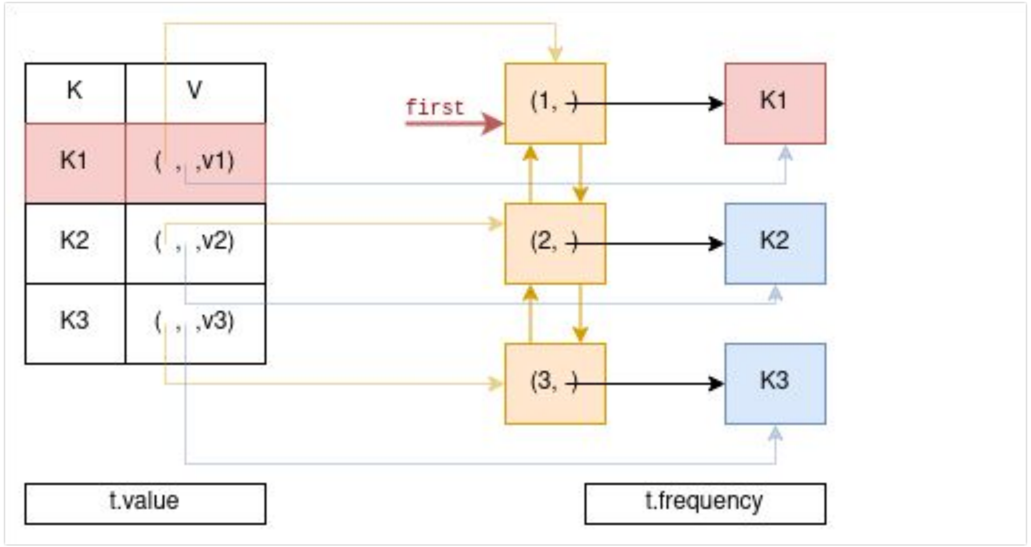
Mem t k3

Cache size = 3



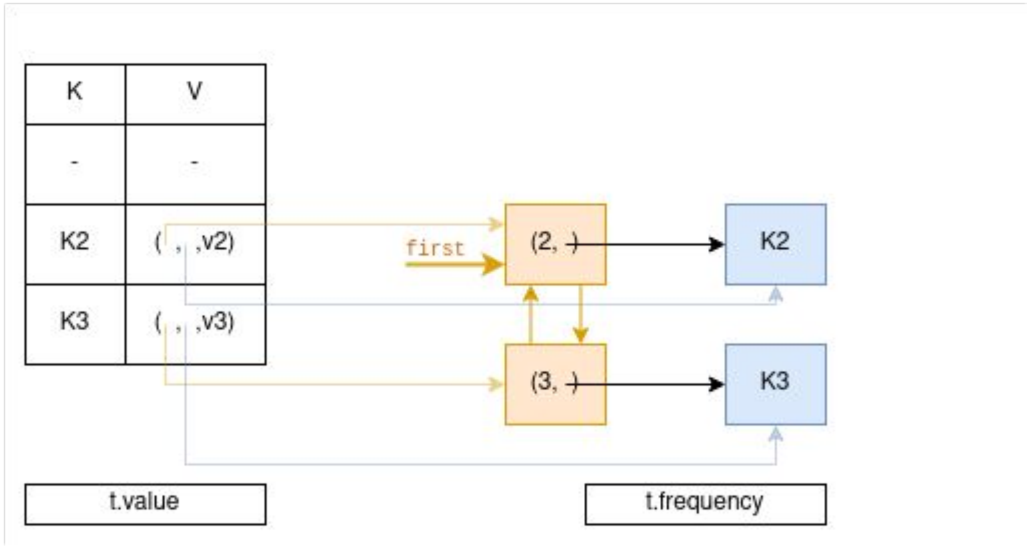
Add t k4 v4

Cache size = 3



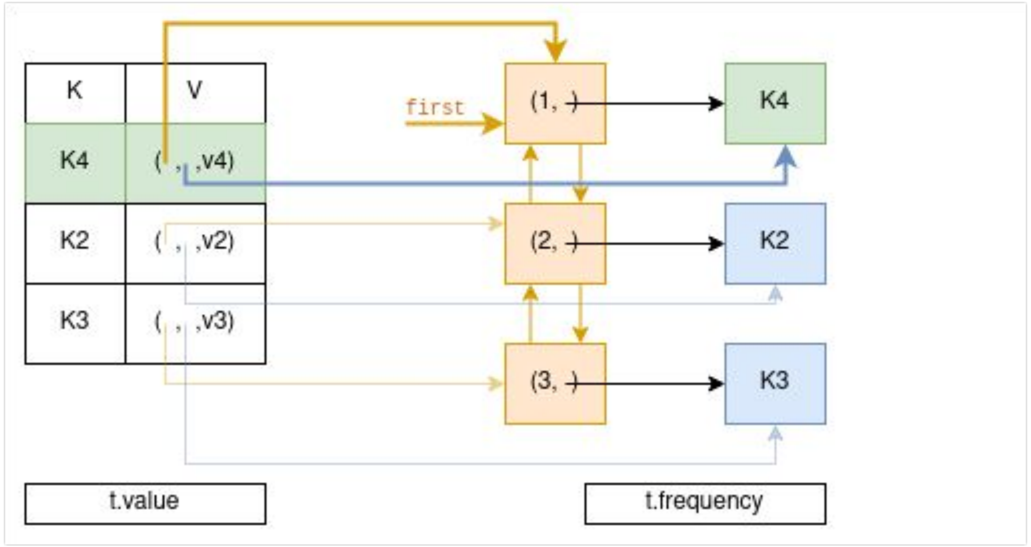
Add t k4 v4

Cache size = 3



Add t k4 v4

Cache size = 3

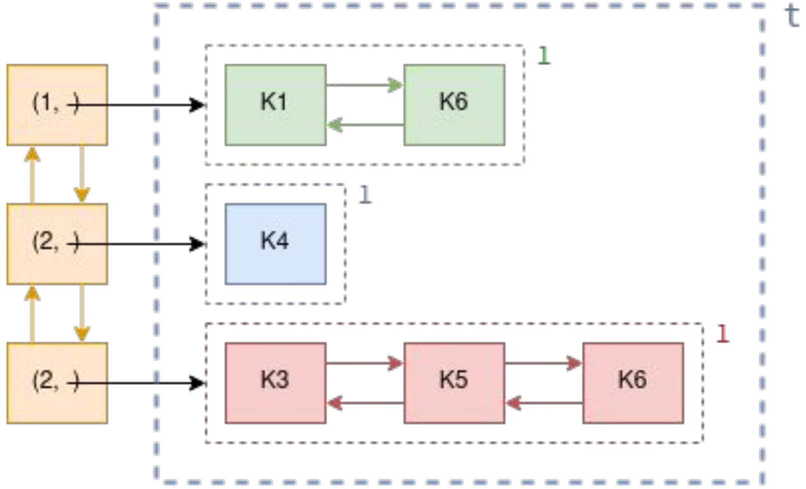


Implementation

Multiple linked lists share the same memory space

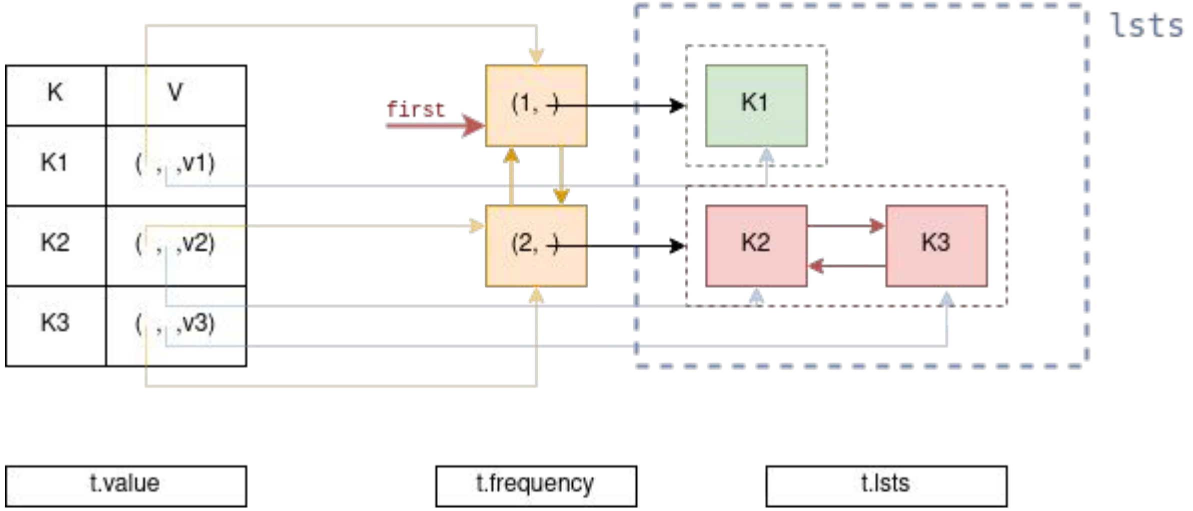
```
type 'a t = {
 cap : int;
 witness : 'a;
 mutable free : int;
 contents : 'a array;
 prev : int array;
 next : int array;
}
```

```
type 'a l = {
 mutable first : int;
 mutable last : int;
 mutable size : int;
 t : 'a t;
}
```



LFU Data-structure

```
type 'a t = {
   cap       : int;
   value     : (key, freq_cell * key_cell * 'a) Hashtbl.t;
   frequency : freq_list;
   lsts      : key Dllist.t;
   stats     : Stats.t;
 }
```



Tests, Benchmarks, and Formal Specifications

Tests

- Unit tests (Alcotest)

```
Testing ‘LFU’.
This run has ID `BMEVX96H'.

  [OK]          int                    0    Mem finds value just added.
  [OK]          int                    1    Mem finds value just added over capacity.
  [..]
Full test results in ‘~/Documents/git_repo/cachecache/_build/_tests/LFU’.
Test Successful in 0.006s. 17 tests run.
```

- Integration tests (Irmin traces)
- Formal specifications and runtime verification

Gospel

- Formal Behavioural Specification language
- Specify types and functions in interfaces
  - Type invariants
  - Function contracts (preconditions and postconditions)
- Special comments starting with @
- More precise than documentation

```
type 'a t = {
 cap : int;
 prev : int array;
 next : int array; (*...*)
}
(*@ with t
    invariant t.cap > 0
    invariant forall i. 0 <= i < t.cap ->
                         t.next.(i) <> -1 ->
                         t.prev.(t.next.(i)) = i
*)
```

Ortac

- Generate OCaml code from Gospel contracts named `*_rac.ml`

```
Dllist -> Dllist_rac dans lfu
```

- Add assertion from conditions which stop the program
- Easy to find bugs
- Provide better bugs report

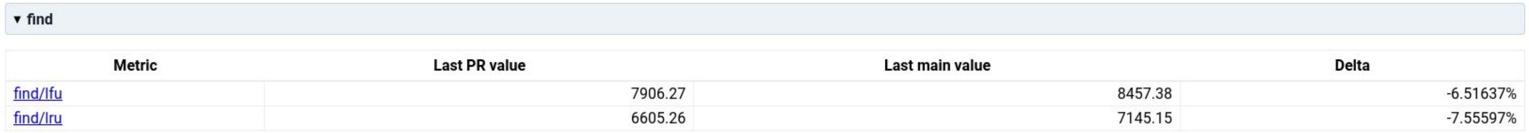
```
$ ./client
File "dllist.mli", lines 15-18, characters 2-43:
Runtime error in function ‘append’:
- the post-condition ‘c = old (l.t.free)’ was violated.
```

Benchmarks

Micro-benchmarks (bechamel)

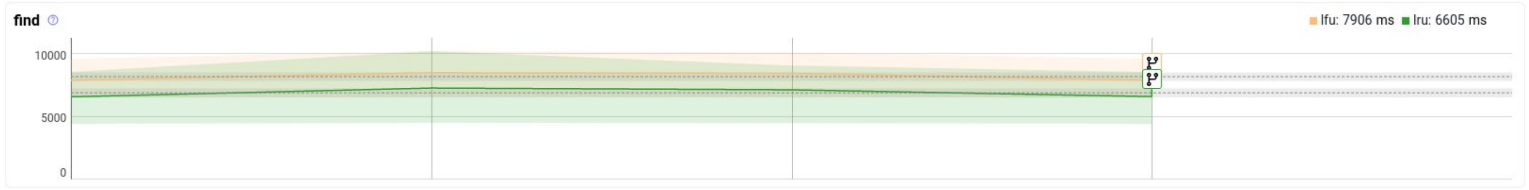
- LFU 1,4 times slower than LRU
- Allocate 2 times more than LRU
- Efficient (allocate 3 times more than Hashtbl)

Integration benchmarks (Irmin traces)



▾ find

Metric	Last PR value	Last main value	Delta
find/lfu	7906.27	8457.38	-6.51637%
find/lru	6605.26	7145.15	-7.55597%



Thank you