Hashes, Hashmaps, frames, structs, ...

Erlang User Conference Stockholm 2011

Björn-Egil Dahlberg, Kenneth Lundin

Background

Erlang/OTP makes the programmer productive

But!

there is always room for improvements

for example

RECORDS

- RECORDS are very frequently used
- Natural choice for representation of structured data
  - few named fields
  - possibly many instances
- Records are declared,
  - have named data fields, with optional default values,
  - can be matched
  - special syntax for records.

Records continued

```
-record(Person, {firstname, lastname, age}).
P1 = #person{firstname="John", lastname="Doe",age=45},
P2 = P1#person{firstname="Carl"},
FirstName = P2#person.firstname,
P3 = P2#person{age = P2#person.age+1},
myfunction(Person = #person{firstname = Firstname}) ->
```

BUT!

- not a distinct datatype (tuples)
- declared in include files (compile time dependencies)
- problems with upgrade
- somewhat inefficient syntax
- not suitable for large number of fields

Hash maps

Associative arrays,

Known from other languages:

- Perl ,hashes,
- Python dictionaries,
- Clojure, maps struct_maps

Typically many elements (keys)

Keys can typically be strings

not so many instances

In Erlang we have `proplists, dict, gb_trees, gb_sets` etc.

Hash maps continued

proplists

```
P1 = [ {firstname, “John”}, {lastname, “Doe”},{age,
 45}],
P2 = [{firstname,”Carl”}|proplists:delete
 (firstname,P1)],
FirstName = dict:fetch(firstname,P2),
P3 = dict:update(age, fun(V) -> V+1 end, P2 ),
```

BUT!

Inefficient memory wise

Bad performance for lookup when many keys.

Can not be matched, is not a data type of its own.

Hash Maps continued

dict

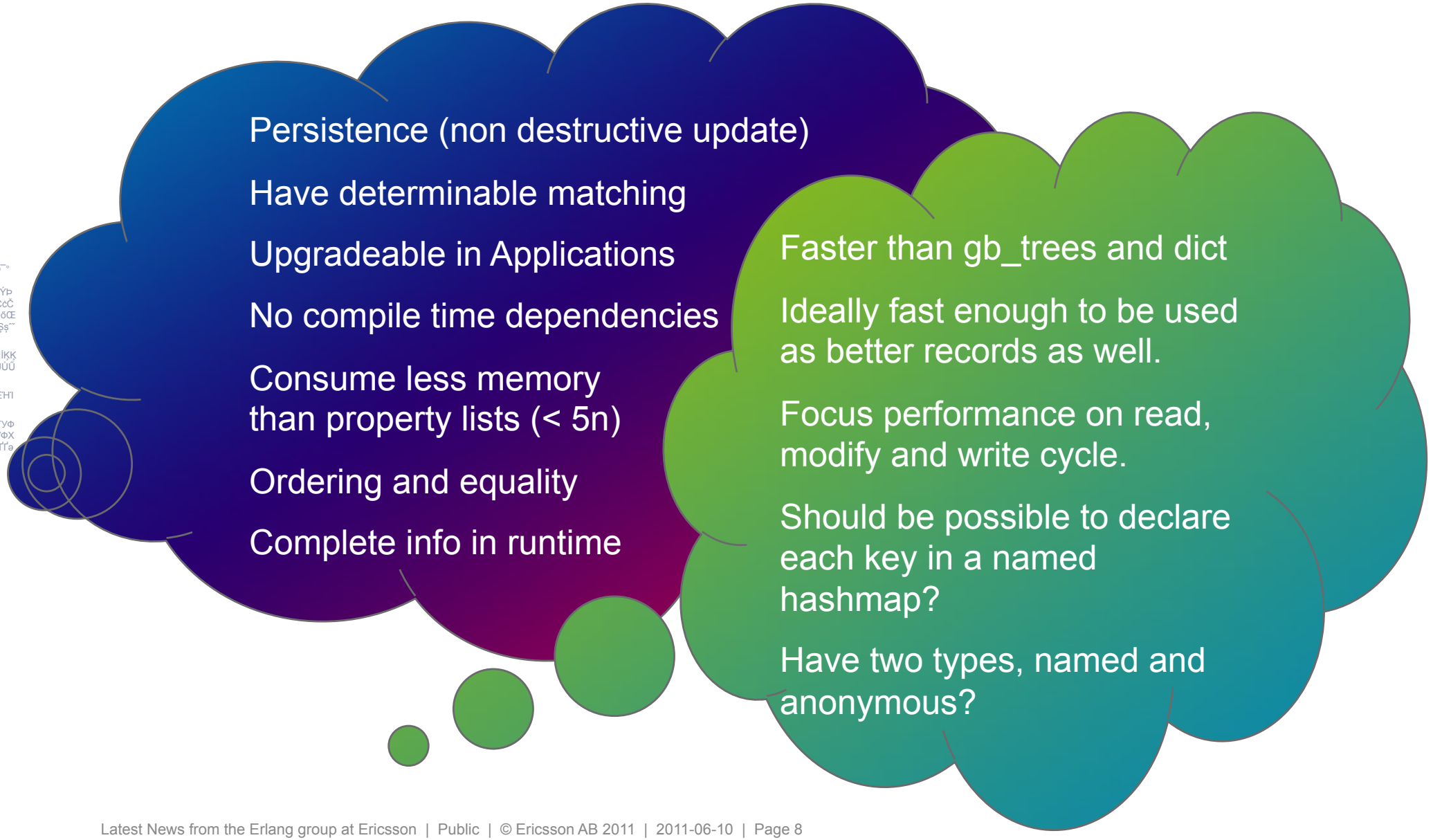
```
P1 = dict:from_list([ {firstname, “John”}, {lastname,
  “Doe”},{age,45}]),
P2 = dict:store(firstname,”Carl”,P1),
FirstName = dict:fetch(firstname,P2),
P3 = dict:update(age, fun(V) -> V+1 end, P2 ),
```

BUT!

can not be matched,

rather inefficient (but better than proplists) to access for large number of keys.

Requirements/Goals HashMaps



Input

- Richard O'Keefes frames proposal
  - record replacement
- Joe Armstrongs struct proposal
  - record replacement
- "Erlson" by Anton Lavrik (https://github.com/alavrik/erlson)
  - syntax similar to records, using dict as representation
- Phil Bagwell hash trees and vlists
  - internal representations

Syntax (some potential solutions)

Similar to records but without the type name (from Erlson)

```
X = #{} % create an empty dictionary
P = #{firstname = "John", lastname = "Doe", age =
 45}.
45 = P.age,
P1 = P#{ firstname = "Carl"},
P2 = P1#{age = P1.age+1},
```

----- OR ----------

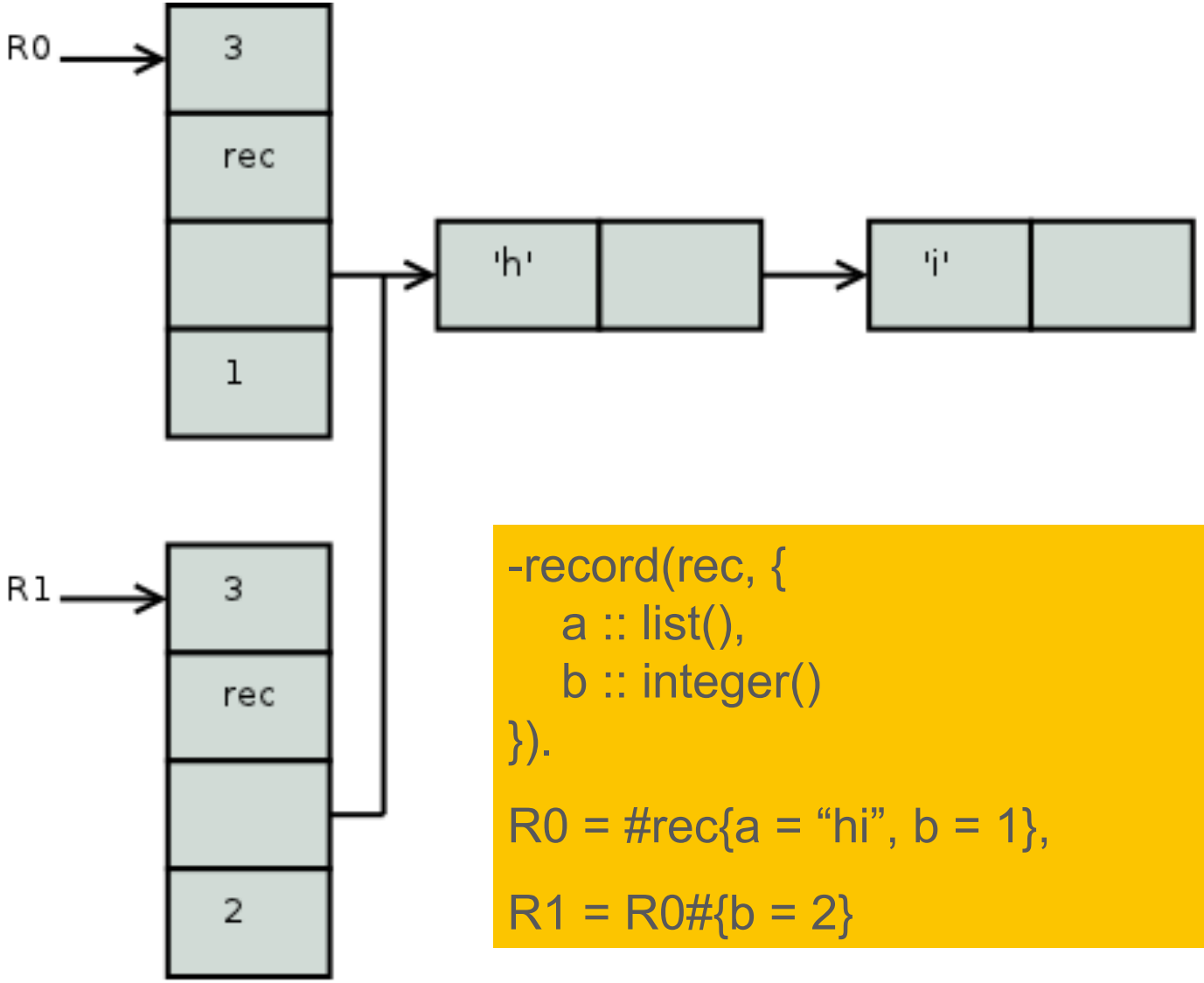
```
45 = P#{age},
["Carl",46] = P2#[firstname,age],
```

Evaluation of Data structures

- We have implemented prototypes and performed measurements using the Vlists and HAMT datastructures internally in the Erlang VM.

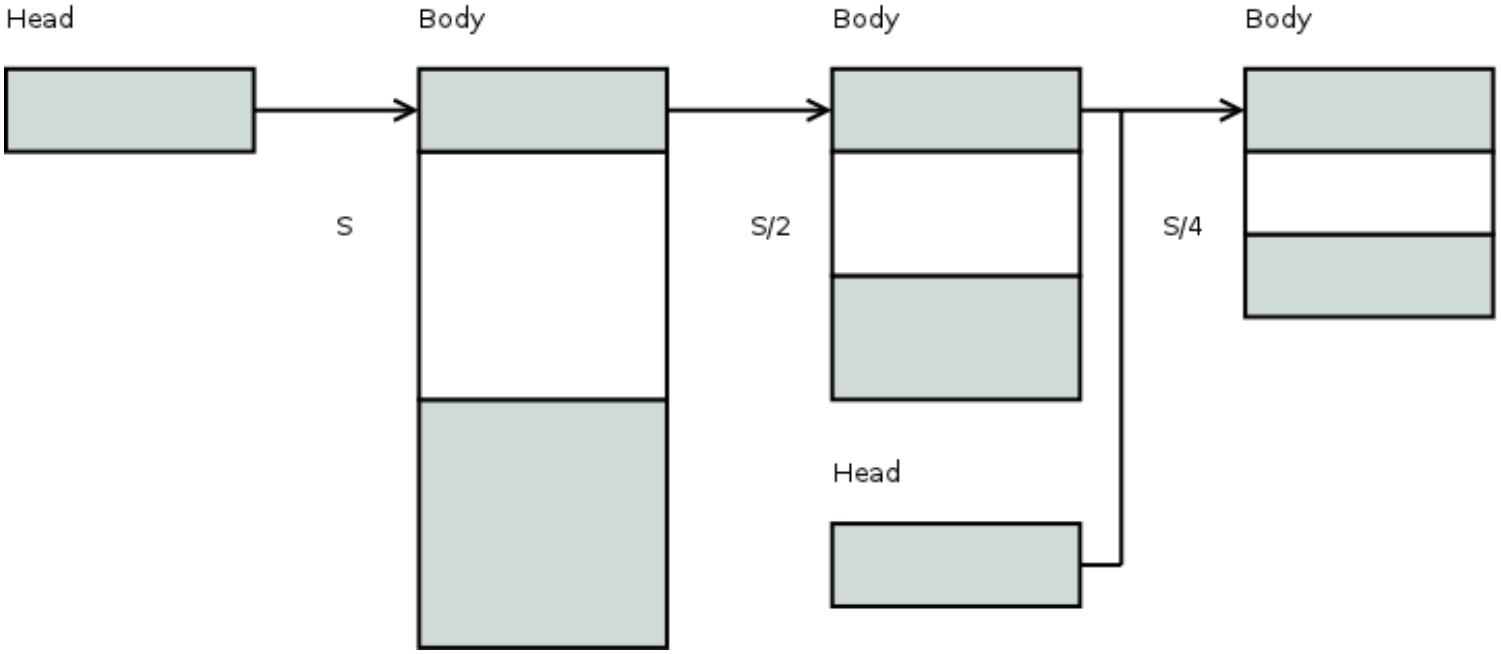
- Records (=tuples)
- Vlists, a linked list of increasingly larger hashtables
- HAMT, Hash Array Mapped Tries
- Hash table with exception lists

Tuples



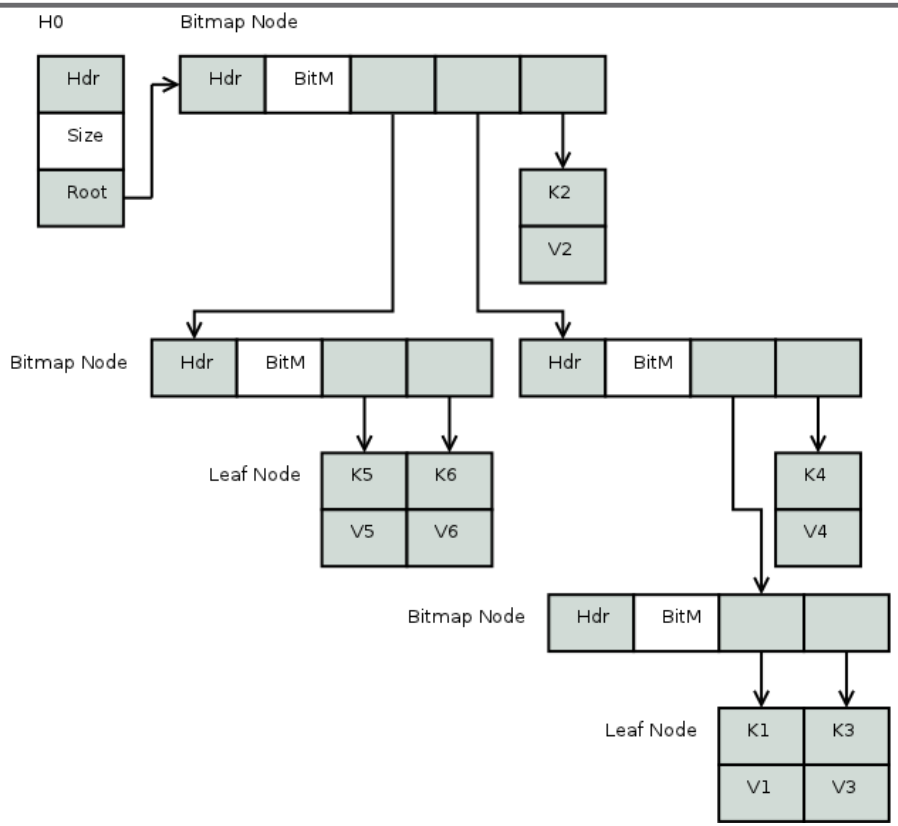

```
-record(rec, {
    a :: list(),
    b :: integer()
}).

R0 = #rec{a = “hi”, b = 1},

R1 = R0#{b = 2}
```

VLISTS



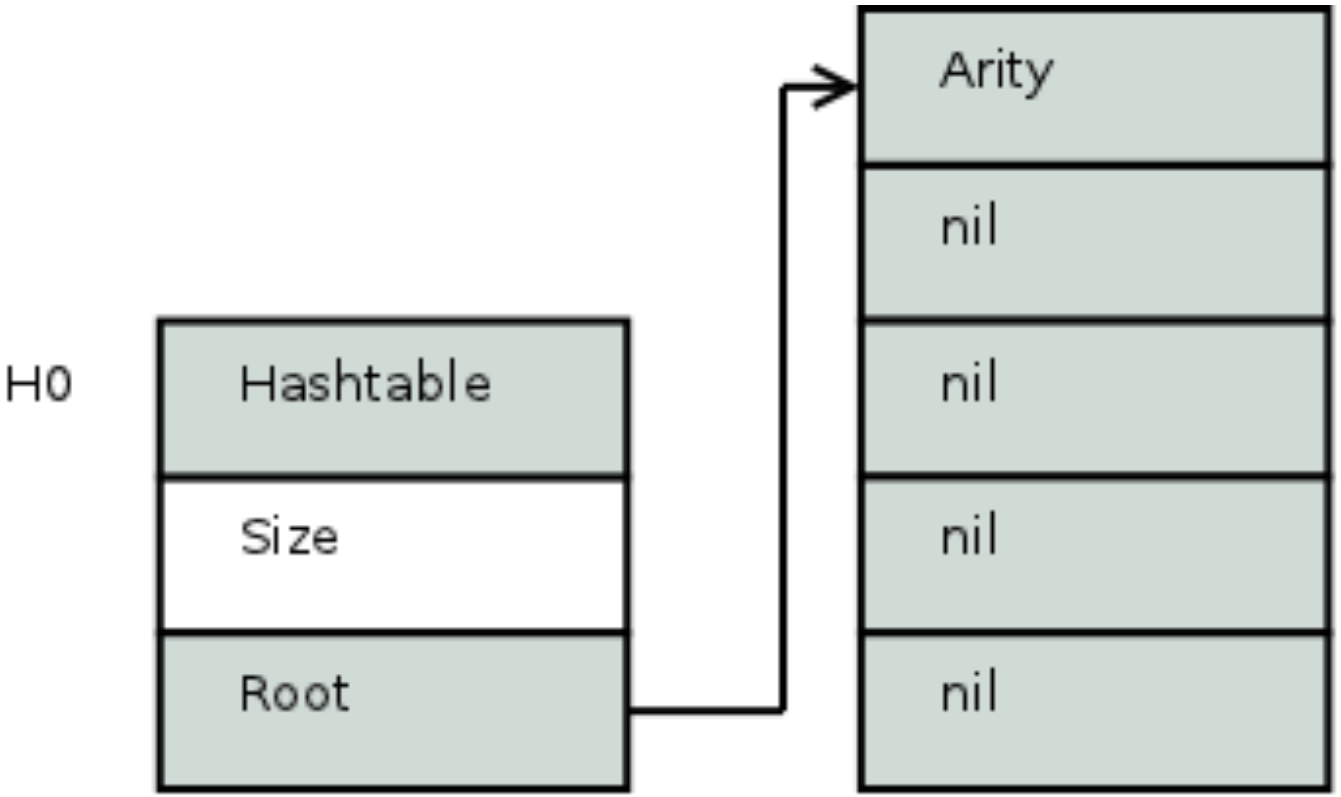
- A vhash list is a linked list of increasingly larger hashtables
- Can be seen as a property list with very fast lookups (O(1))
- Update and delete is expensive

Hamt (Hash Array Mapped Tries)

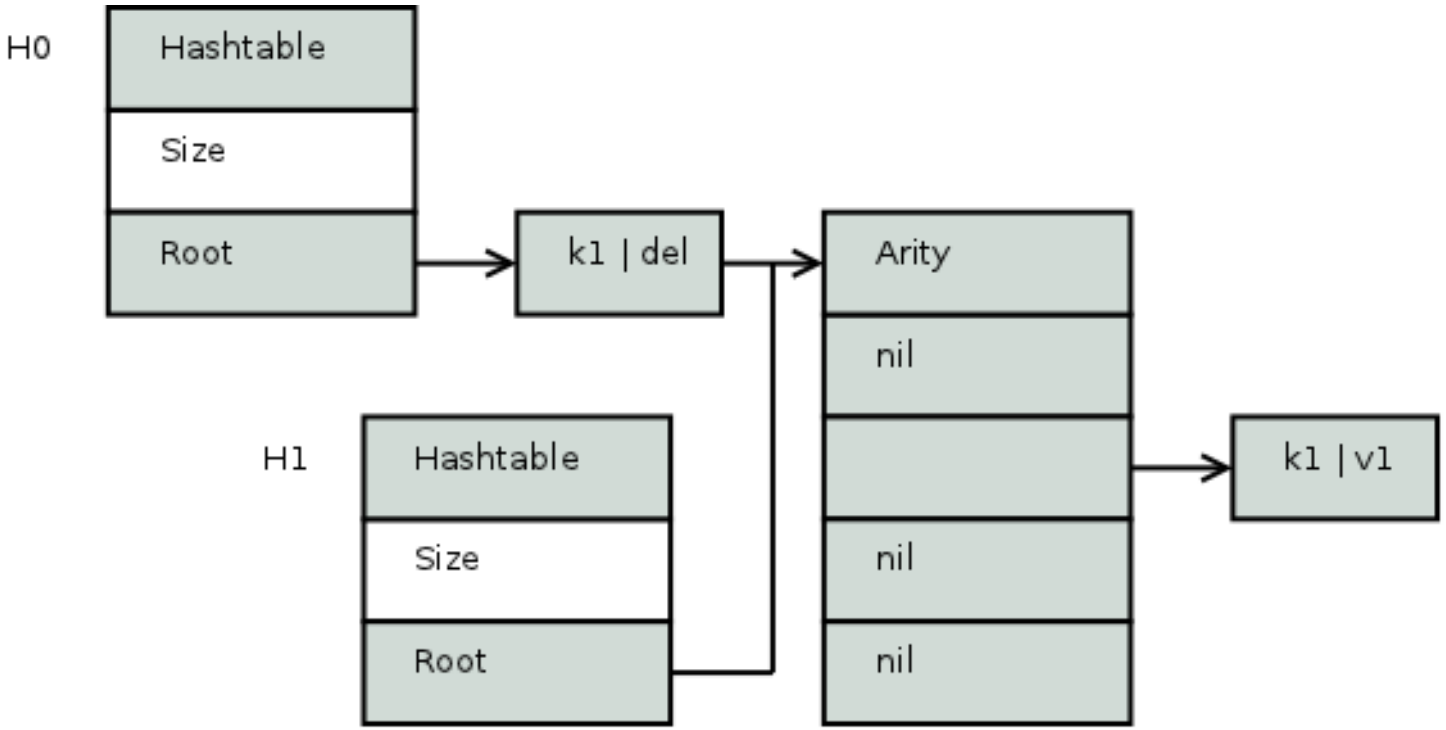


- A shallow and memory compact tree (thanks to the bitmaps)
- bitcount instruction in the processor is very useful for this.
- O(log32) for lookup and updates, (almost as fast as a hash table for reasonable sizes)
- Used for hashmaps in Clojure

Hash table with Exception list

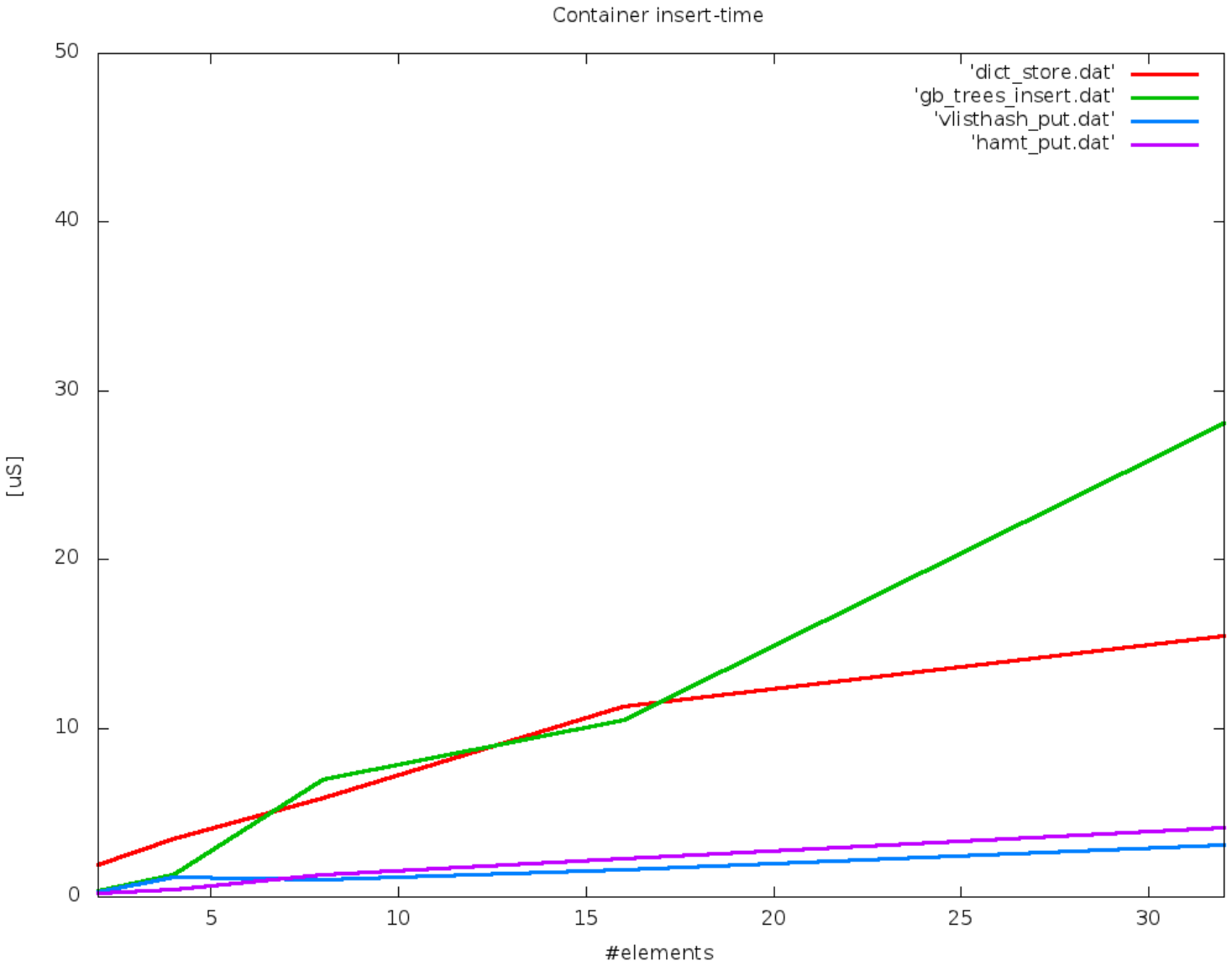


Hash table with Exception list update

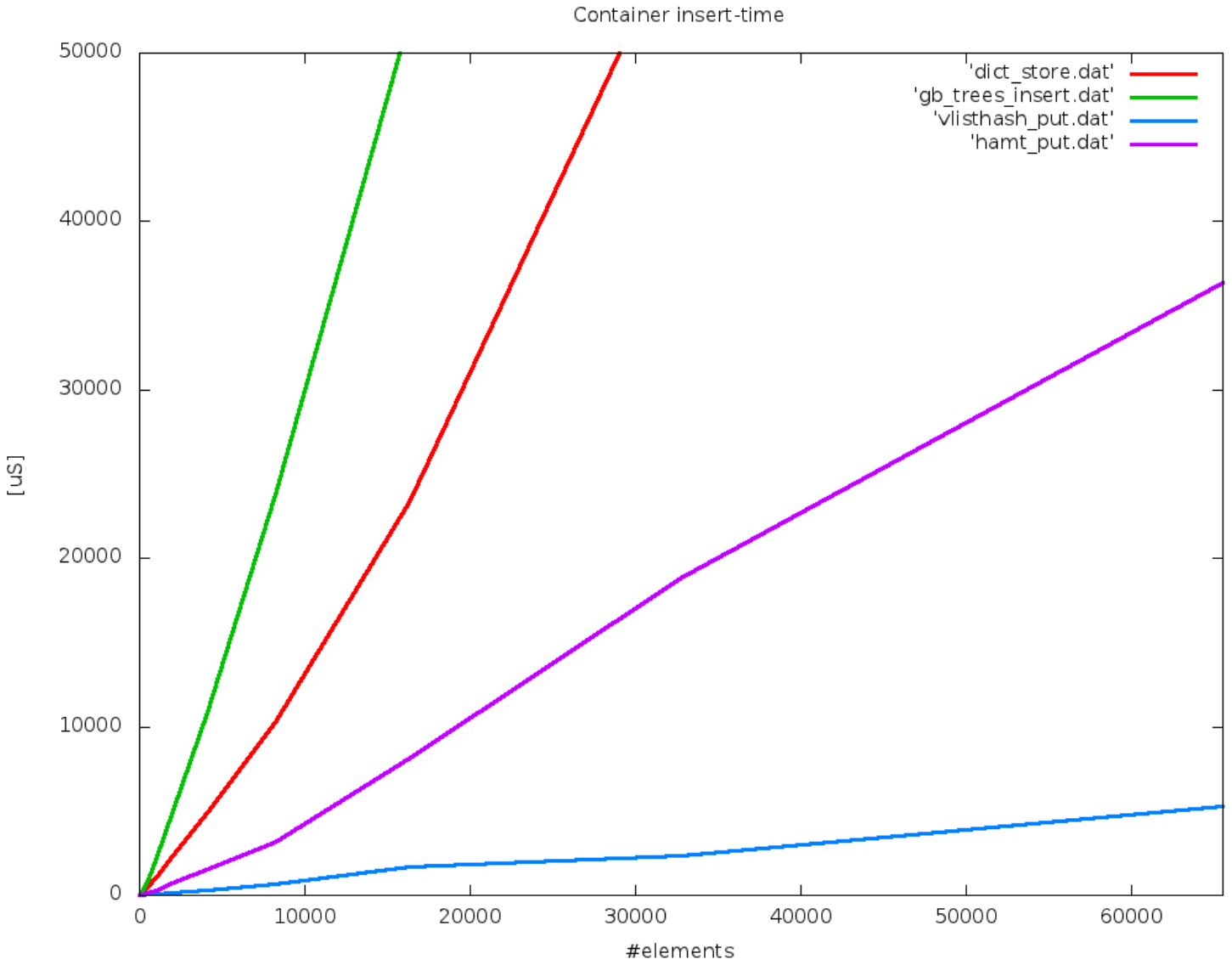


- A hash table which prioritize the latest version
- Older versions gets an additional cost when accessing (traversing the exception lists)

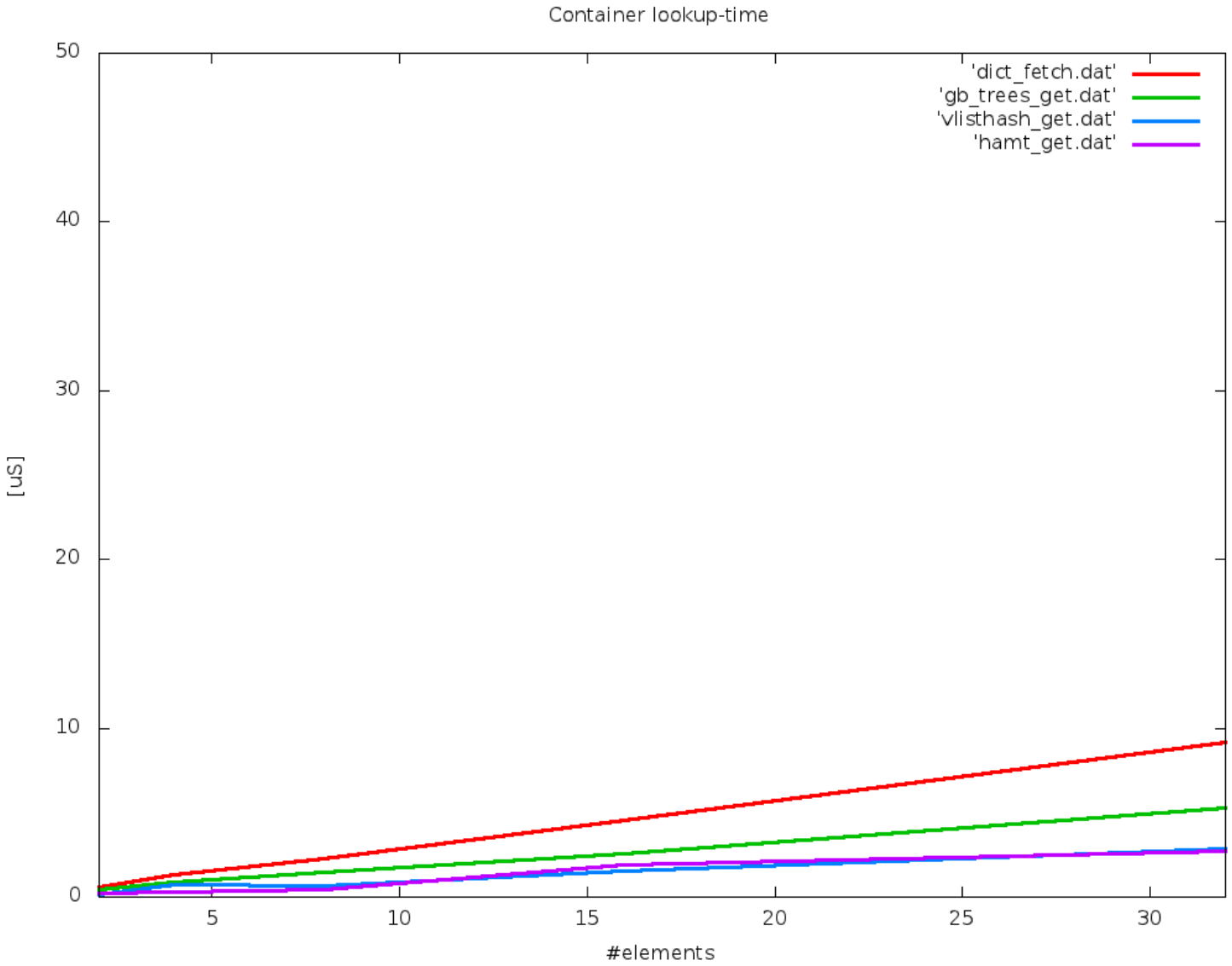
Measurements (insert , small)



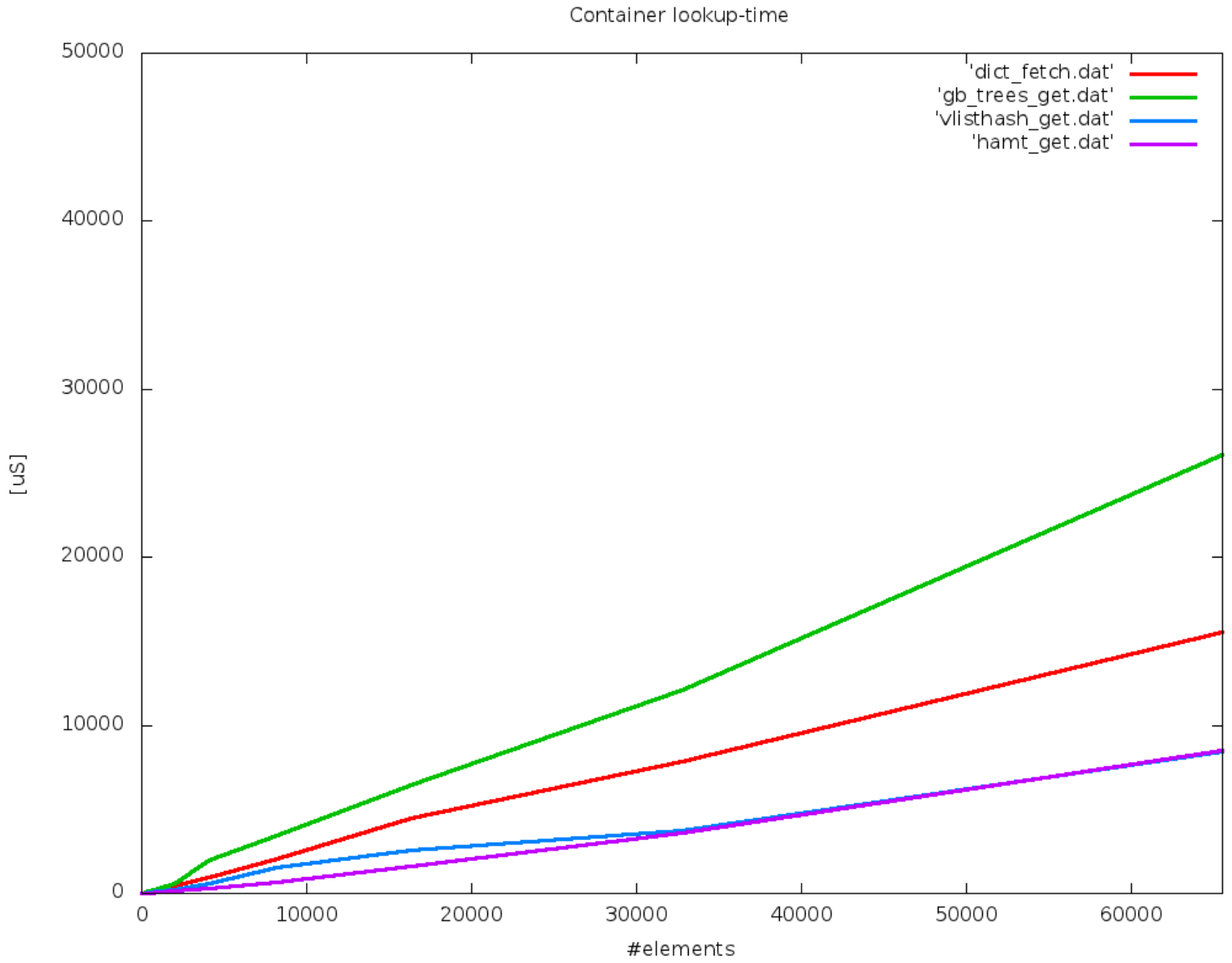
Measurements (insert)



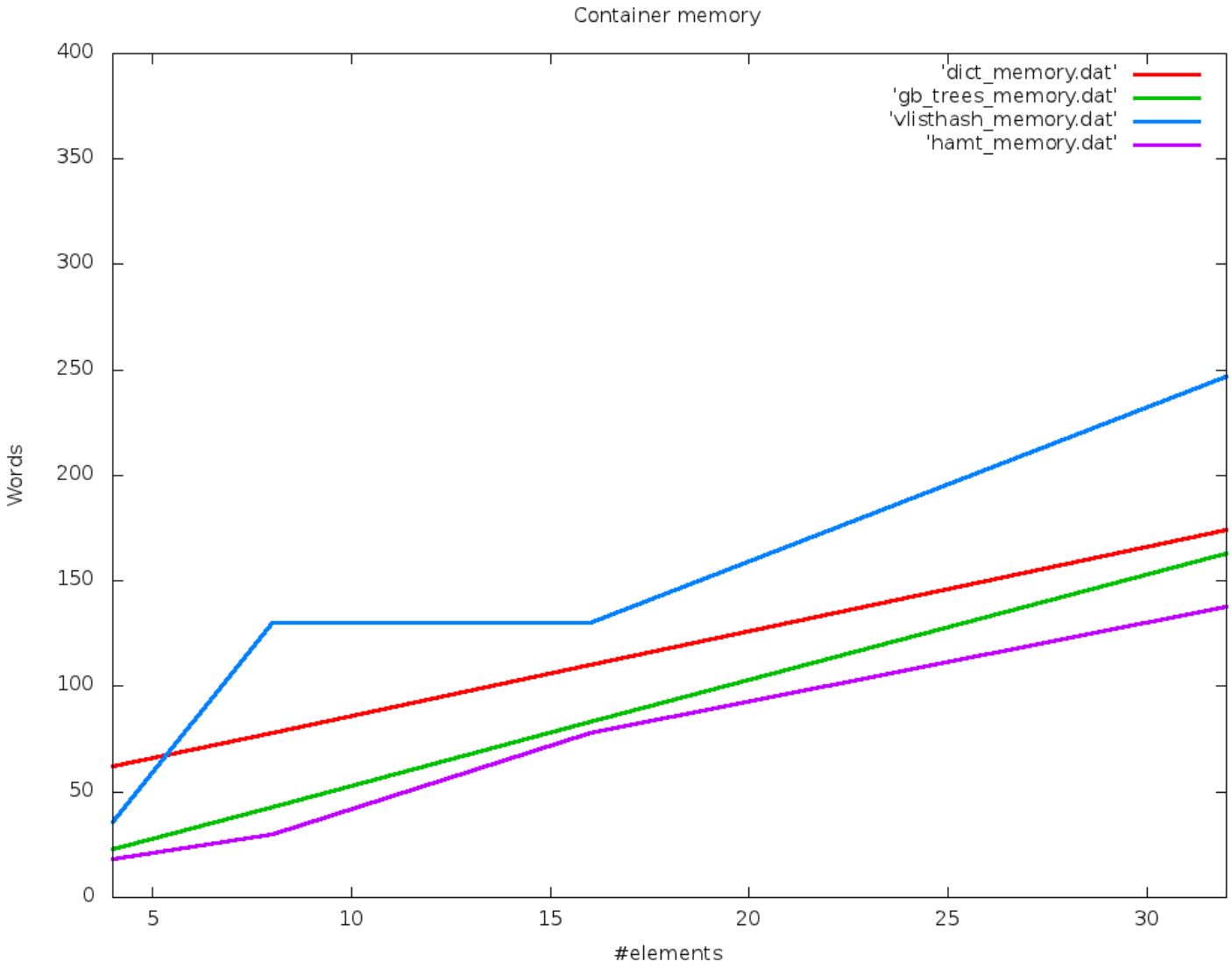
Measurements (Lookup small)



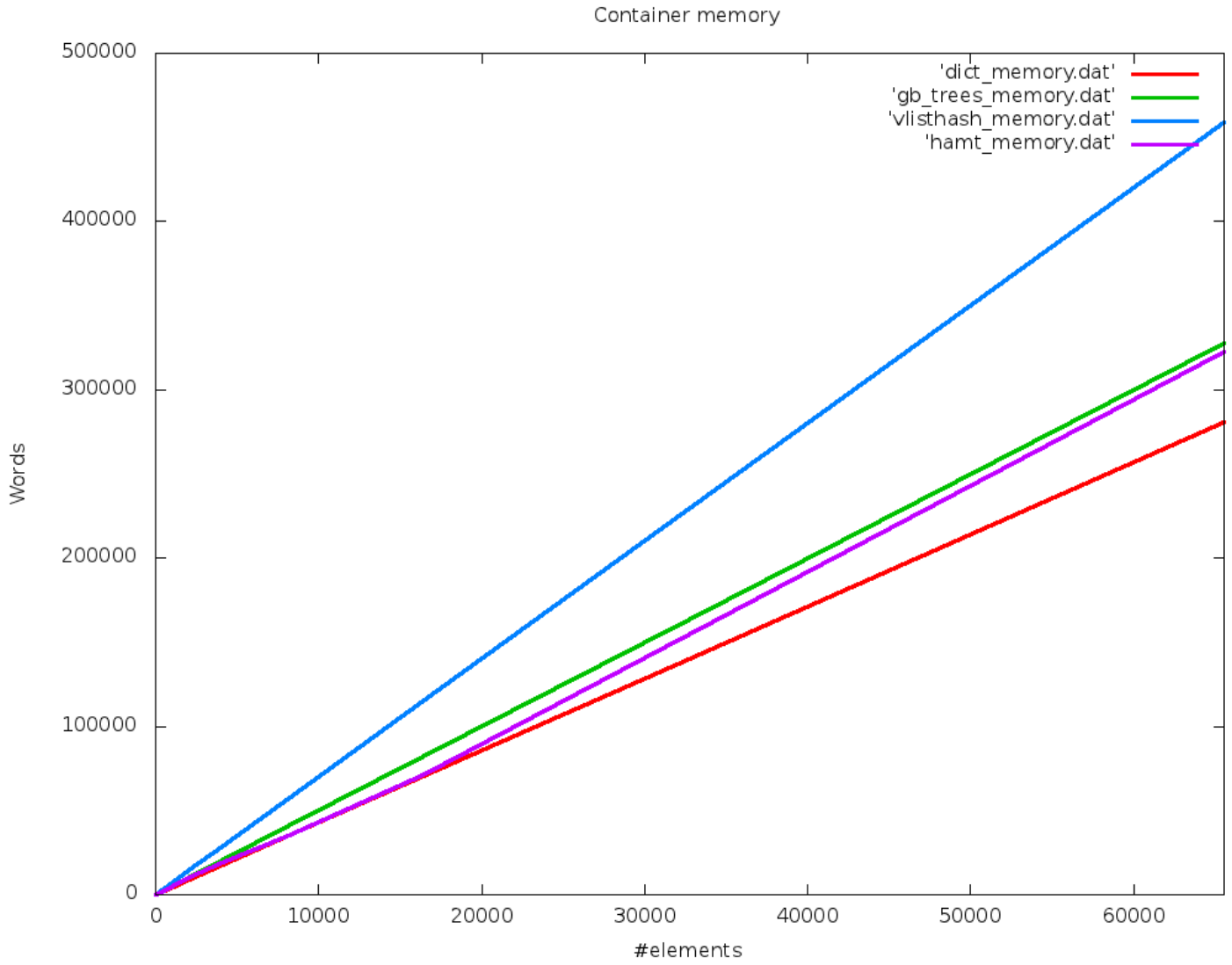
Measurements (Lookup)



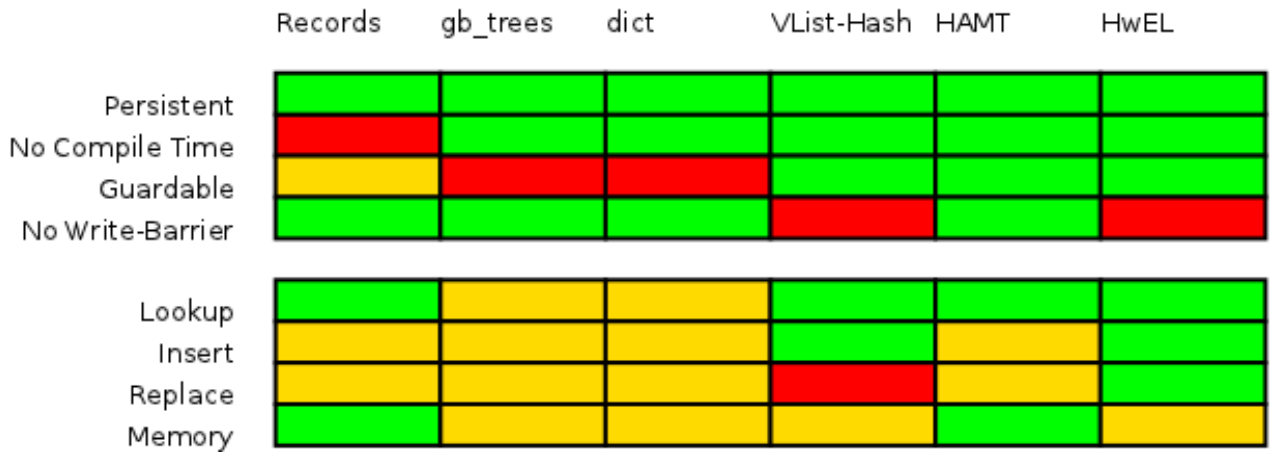
Measurements (memory small)



Measurements (memory)

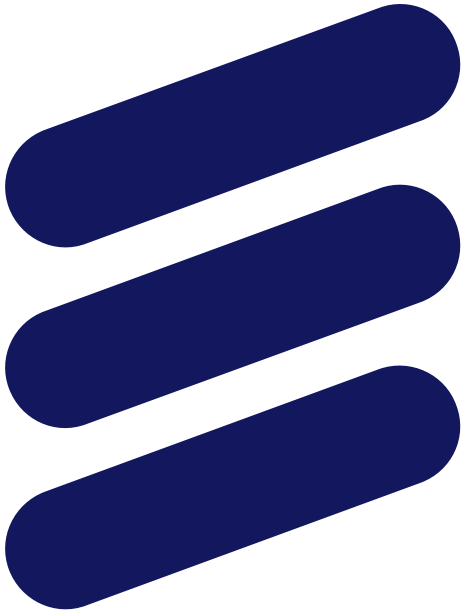


Comparision of requirement fulfillment



Conclusions and way forward

- New better records
  - optionally named, possibly declared
  - few named fields
  - many instances
- ------- and ----------
- Hashmaps
  - many ”keys”
  - few instances
- are two different things (but similar)
- we want both
- HAMT looks really promising for Hash Maps
- but it might be hard to combine with one single representation
- We have to decide what to address first
- Probably some experimental implementation released during 2012



ERICSSON