EFOP-3.6.2-16-2017-00013

European Union

TOWARDS AN ENERGY EFFICIENT COMPUTATION IN ERLANG

Áron Attila Mészáros
Gergely Nagy

Melinda Tóth
István Bozó

SZÉCHENYI 2020

HUNGARIAN GOVERNMENT

European Union
European Social Fund

INVESTING IN YOUR FUTURE

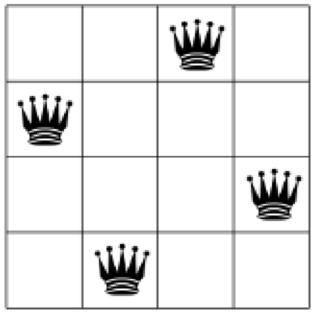
Motivation

- Environmental awareness
  - Energy conservation, minimizing energy consumption
  - Even in computers
  - Green computing

- Why Erlang?
  - Popular languages – lots of research (e.g. C++, Java, Haskell)
  - Erlang
    - Widely used in the industry
    - No such research yet

Previous work

- Tool for measuring energy consumption
- Measuring complex algorithms
  - N-queens
  - Sparse matrix multiplication
- Special attention to:
  - Lists vs arrays
  - Higher order functions
  - Parallelization (parallel map, token ring)



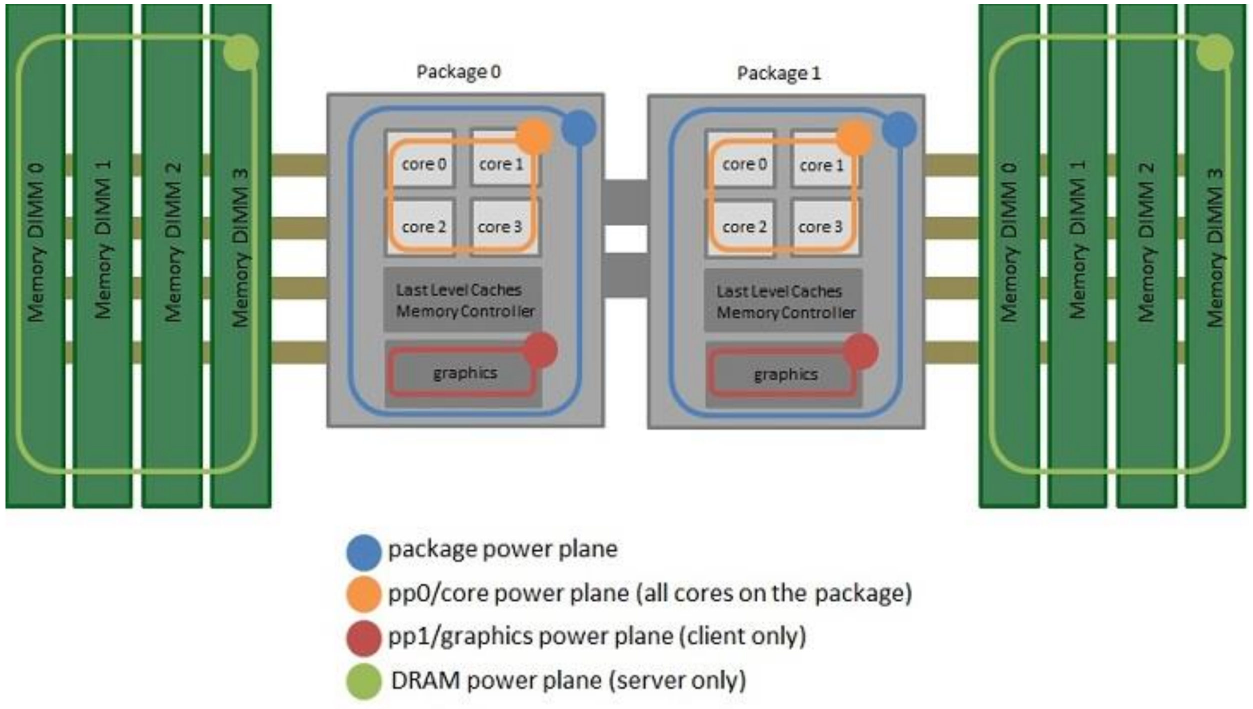
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 5 & 3 & 0 & 7 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 15 & 0 & 16 & 0 \\ 15 & 9 & 0 & 21 \\ 0 & 0 & 12 & 0 \end{bmatrix}$$

New goals

- Improve methodology
- Directly measure language elements
- Measure data structures independently
- Further inspection of parallelization
  - Spawning processes
  - Sending messages
- Refactoring in order to minimize energy consumption

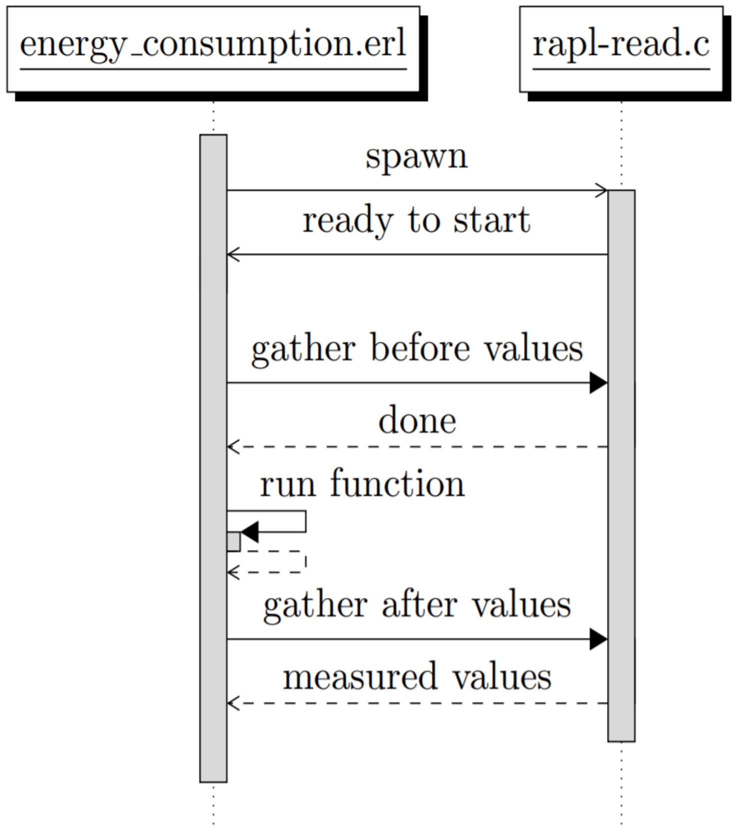
How to measure energy consumption

- Running Avarage Power Limit (RAPL)
  - Intel
  - Methods:
    - MSR
    - perf_event,
    - sysfs
  - Domains:
    - PKG – package
    - PP0 – core
    - PP1 – uncore
    - DRAM



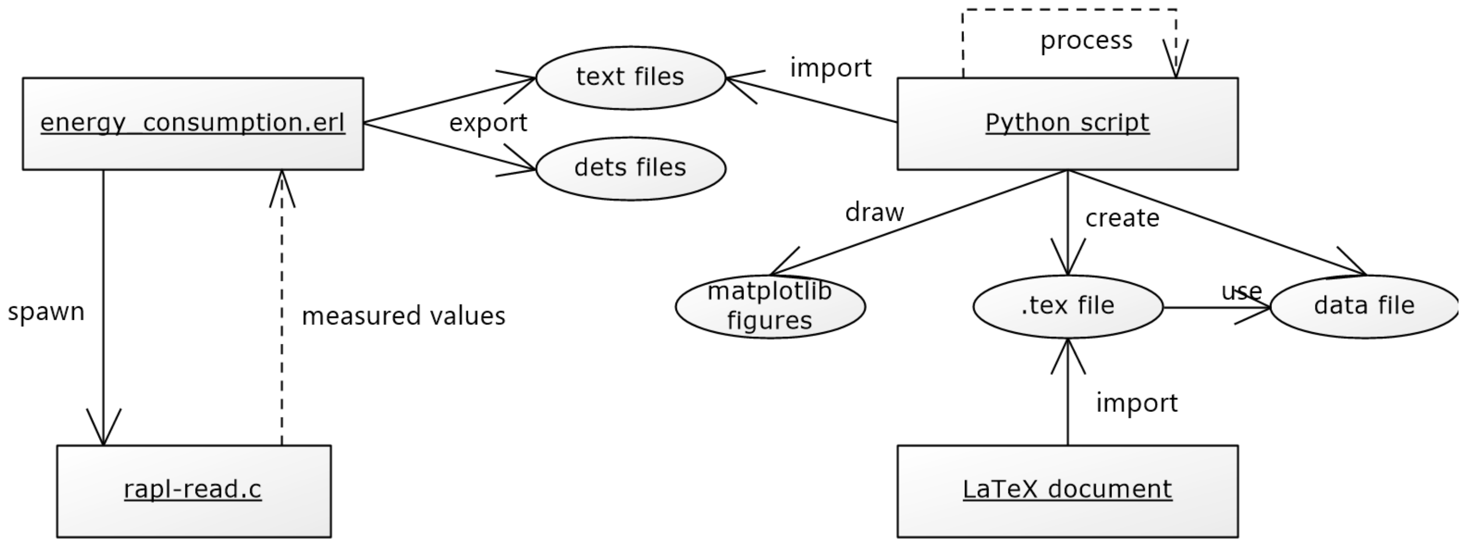
Framework for Erlang

- C program to read RAPL values
- Erlang module to run measured functions
  - Communication via ports
  - Runs the functions
  - Tells the C program when to measure
- Python GUI
  - Using TkInter
  - Setup measurement
  - Helps with organizing the measurements
  - Visualizing results



Methodology

- Methodology
  - Everything is measured 10 times
  - Discarding minimal and maximal values
  - Calculate average energy consumed
  - Also measuring runtime



Measurements

- Higher order functions
  - Important part of functional languages
  - We wanted to find the cost of passing functions as parameters
- Data structures
  - List, map, dictionary
  - What is the cost of creating these data structures?
  - What is the cost of operations?

Higher order functions

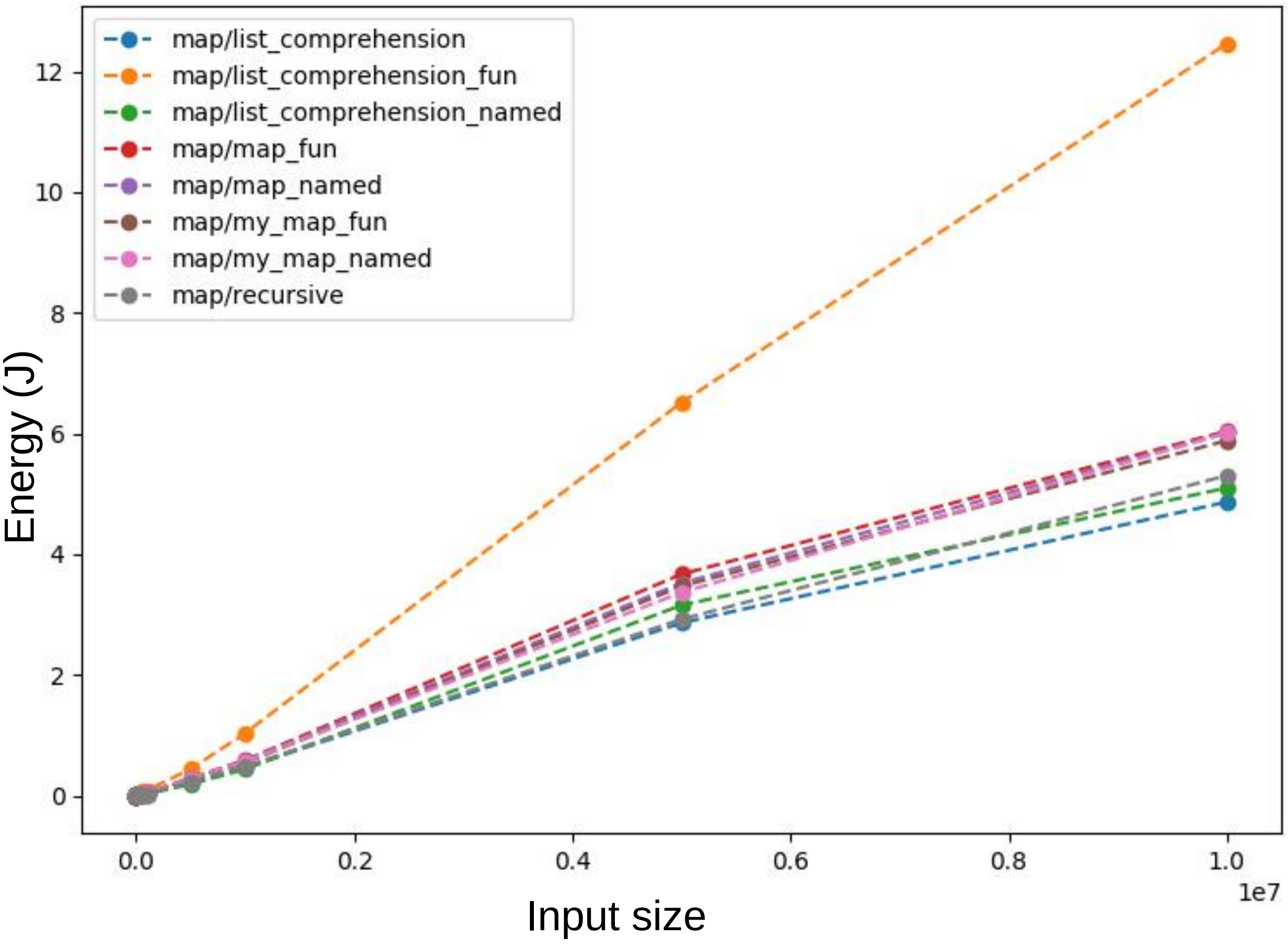
- Passing named or unnamed functions
- Using HOF, list comprehension or recursion
- Implementing our own higher order functions
- Measured:
  - map
  - filter
  - map ∘ filter

```
lists:map(fun increase/1, L).
```

```
lists:filter(fun even/1, L).
```

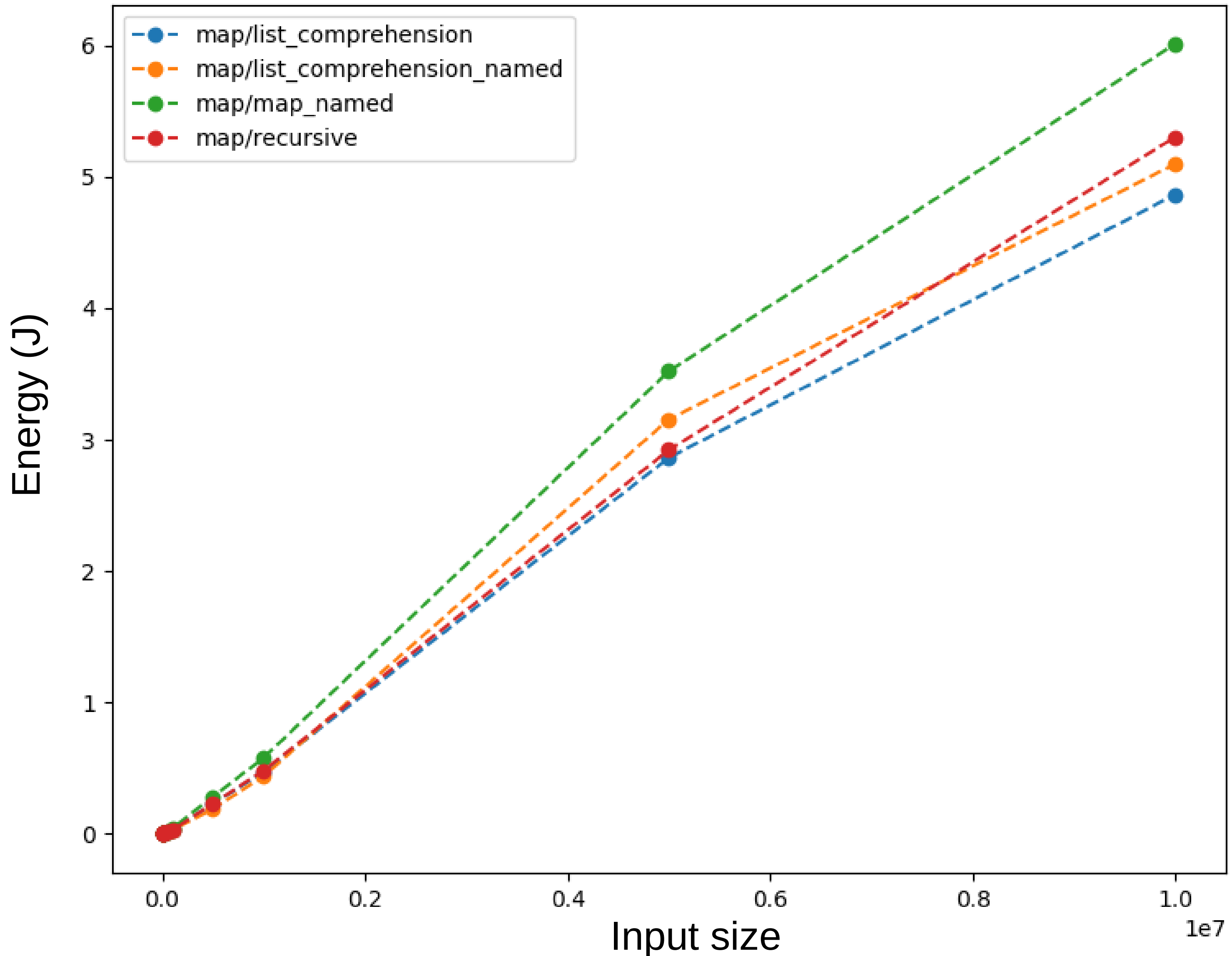
```
lists:map(fun increase/1, lists:filter(fun even/1, L)).
```

Map - results

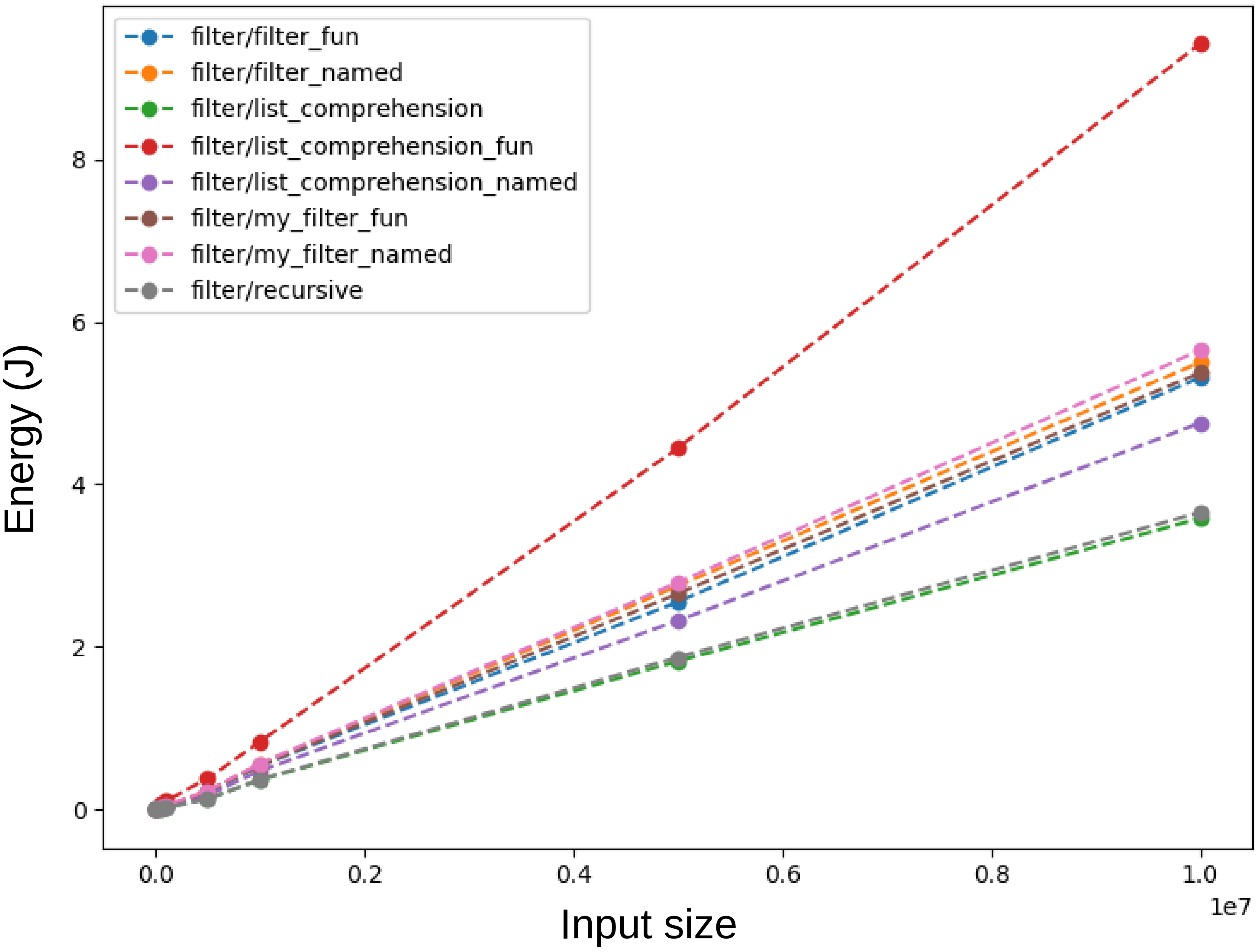


- Best
  - Recursion
  - List comp.
- Worst
  - List comp. with fun.
  - All maps
- Graph is the same based on energy and runtime

MAP - RESULTS

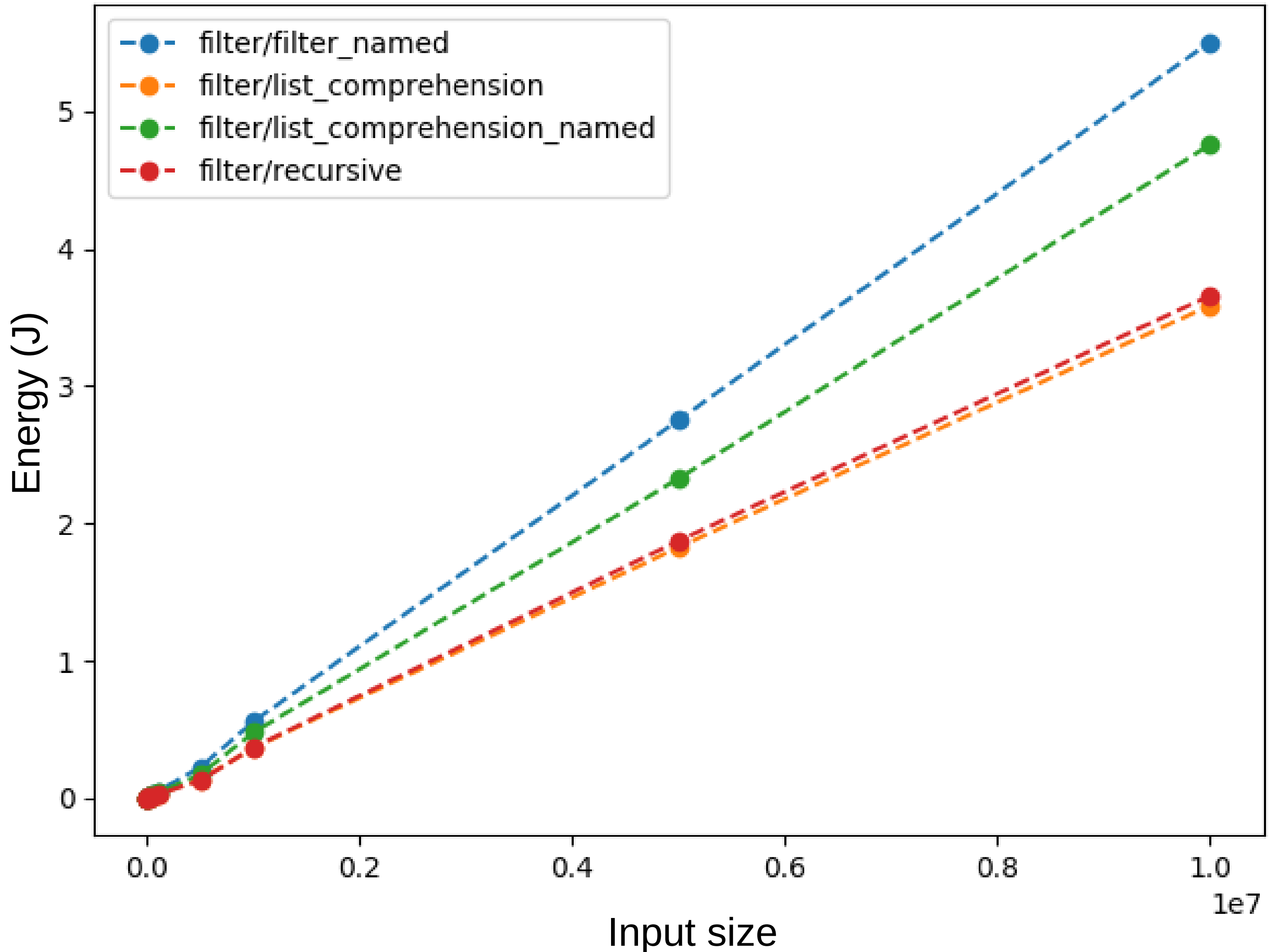

map/list_comprehension
map/list_comprehension_named
map/map_named
map/recursive
Energy (J)
Input size
1e7
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0

FILTER - RESULTS

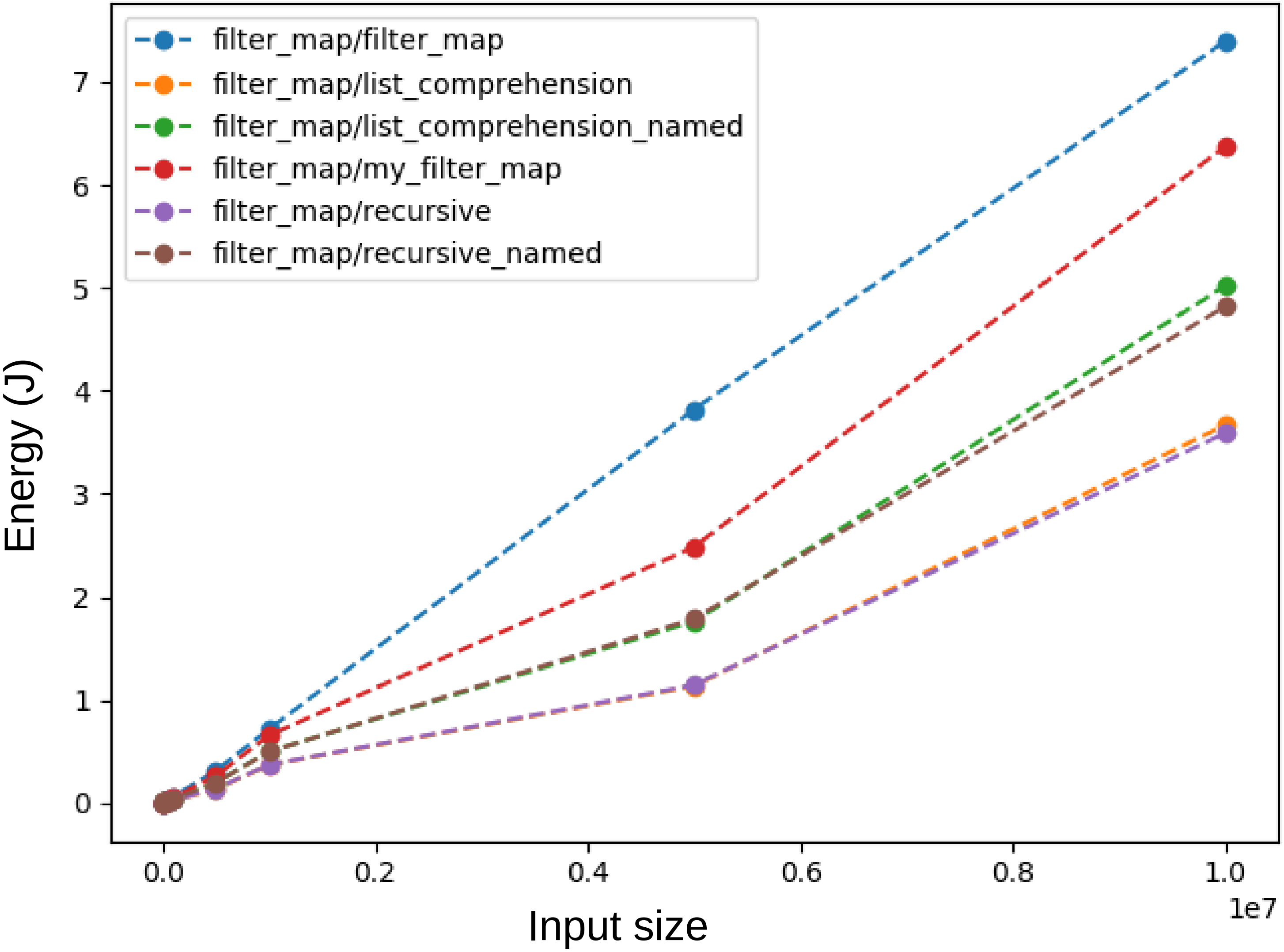


- Same results as with map

Filter - results



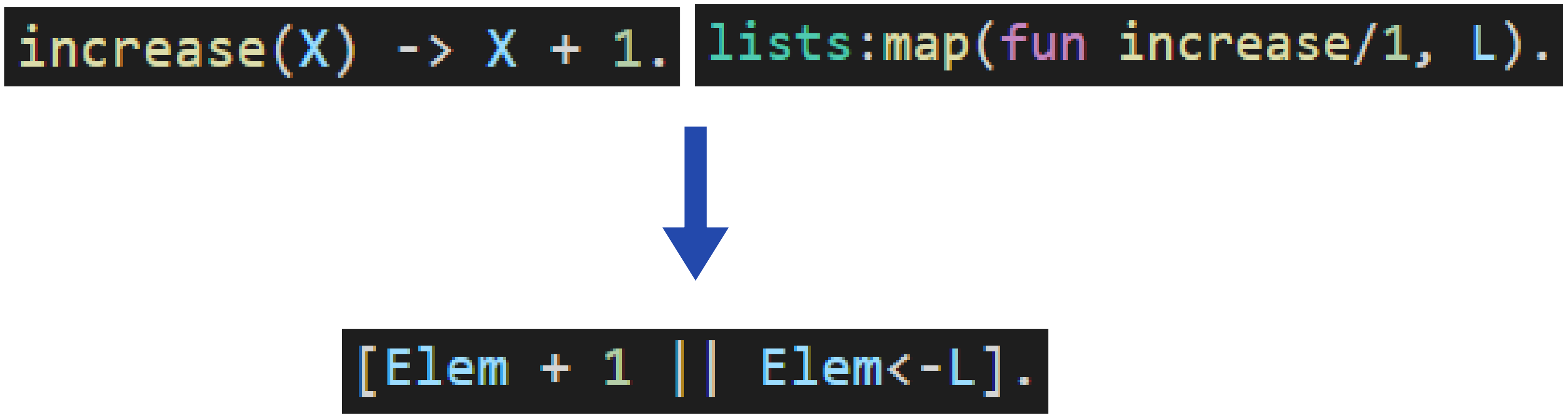
Map ∘ Filter - Results



- Larger differences
- HOF iterates over the list twice

HOFs - FINDINGS

- Refactoring higher order function calls to a simple recursive function or list comprehension reduces energy consumption and runtime
- Eliminating the HOF is not enough, we need to eliminate the called function as well to minimize energy consumption.
- This is consistent with previous findings (N-queens)



```
increase(X) -> X + 1.
```

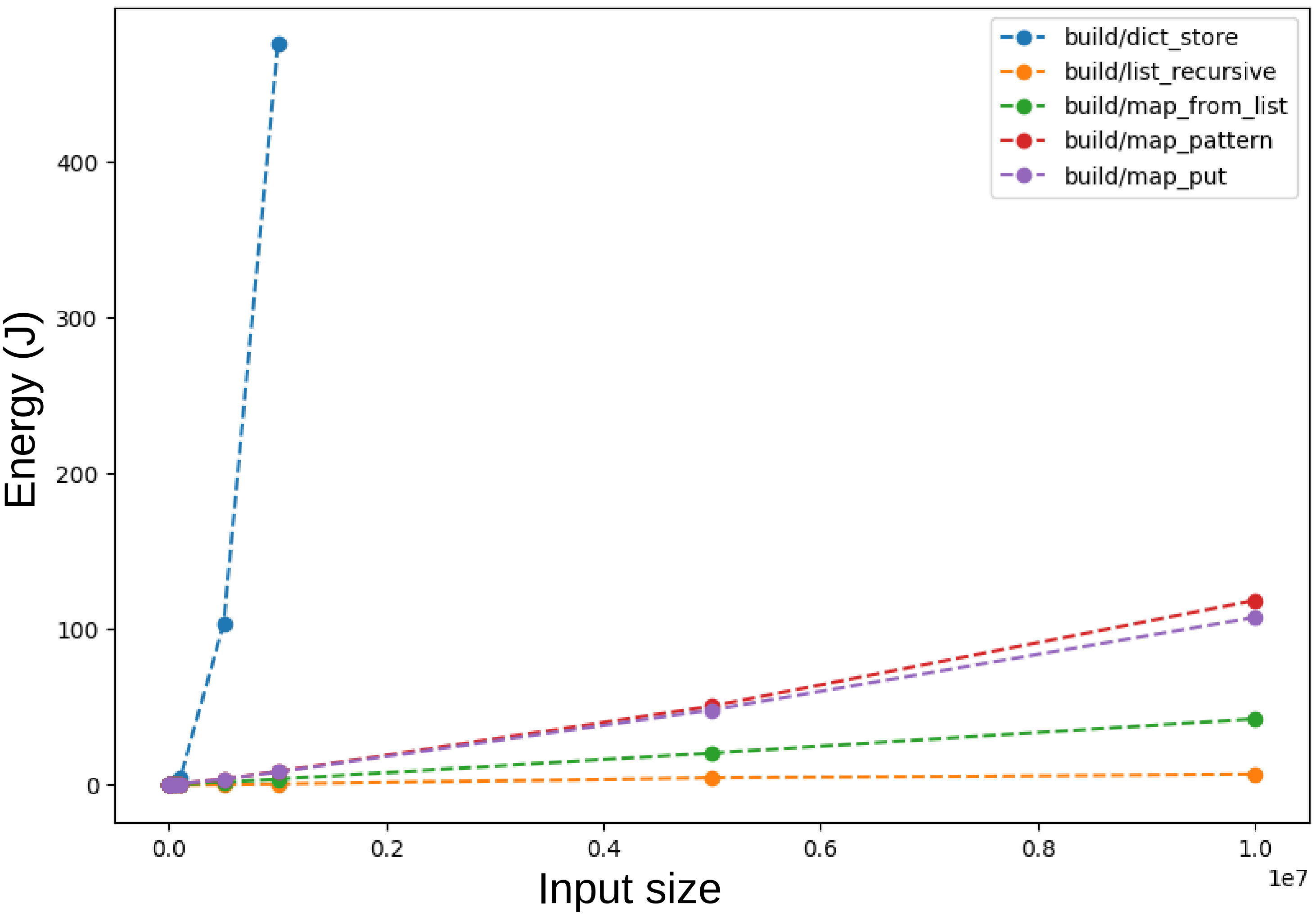
```
lists:map(fun increase/1, L).
```

```
[Elem + 1 || Elem<-L].
```

Data structures

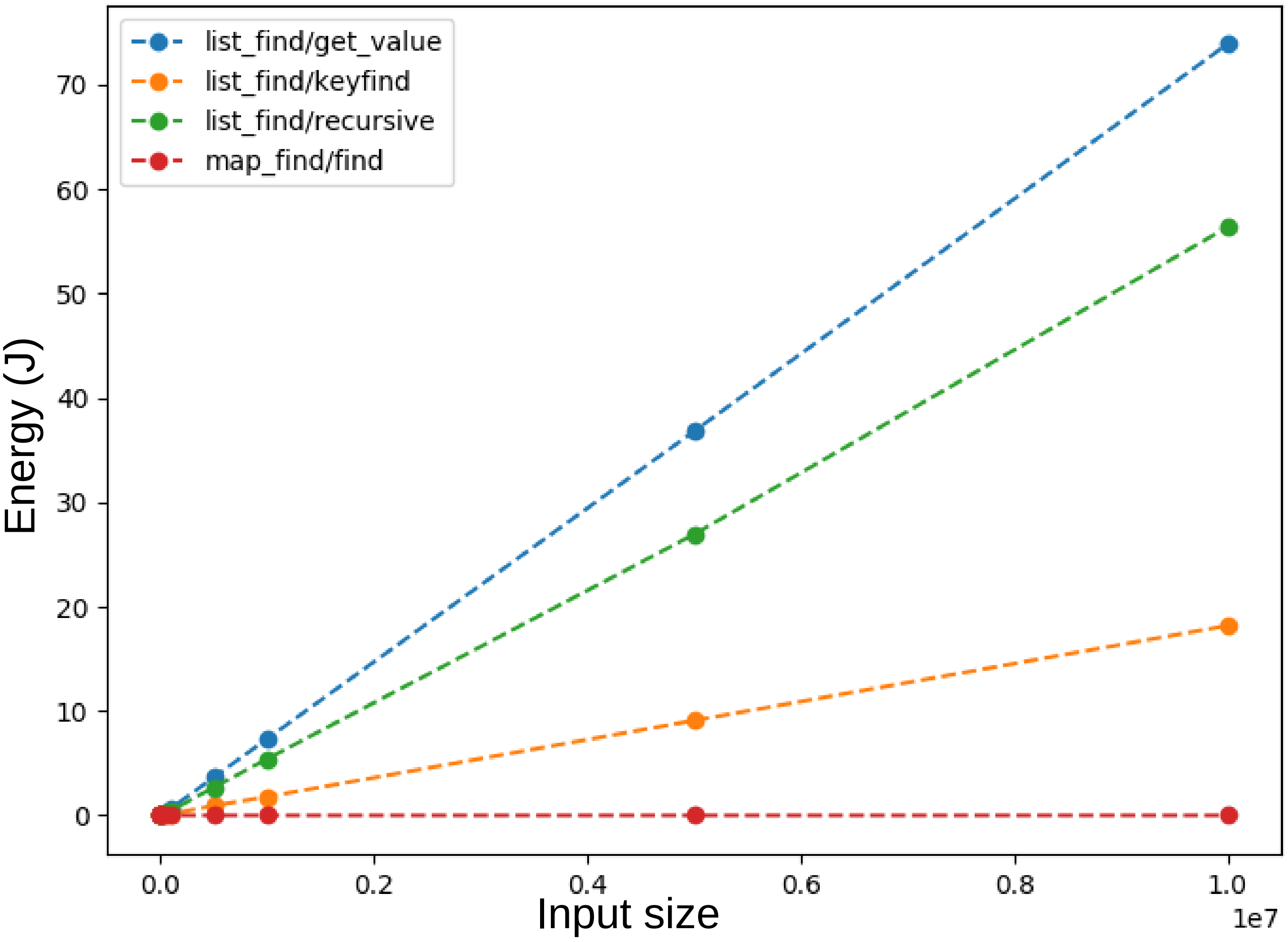
- Investigating different ways to store key-value pairs
  - list of tuples
  - map
  - dict
- Measuring different operations on those data structures
  - find
  - delete
  - update
  - building the data structure

BUILDING - RESULTS



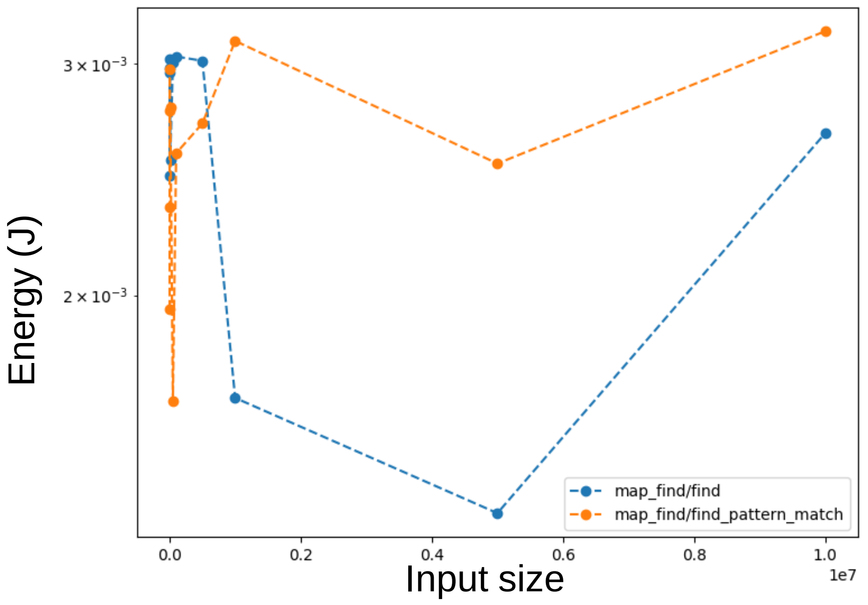
- Dicts are expensive
- Lists are cheap
- Building a list then map is cheaper than building map directly

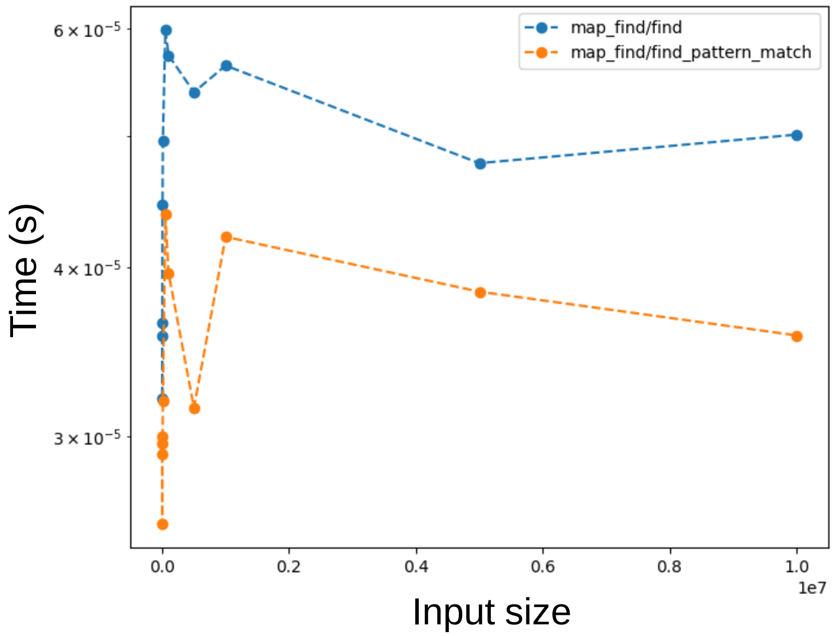
FIND - RESULTS



- proplists:get_value vs lists:keyfind

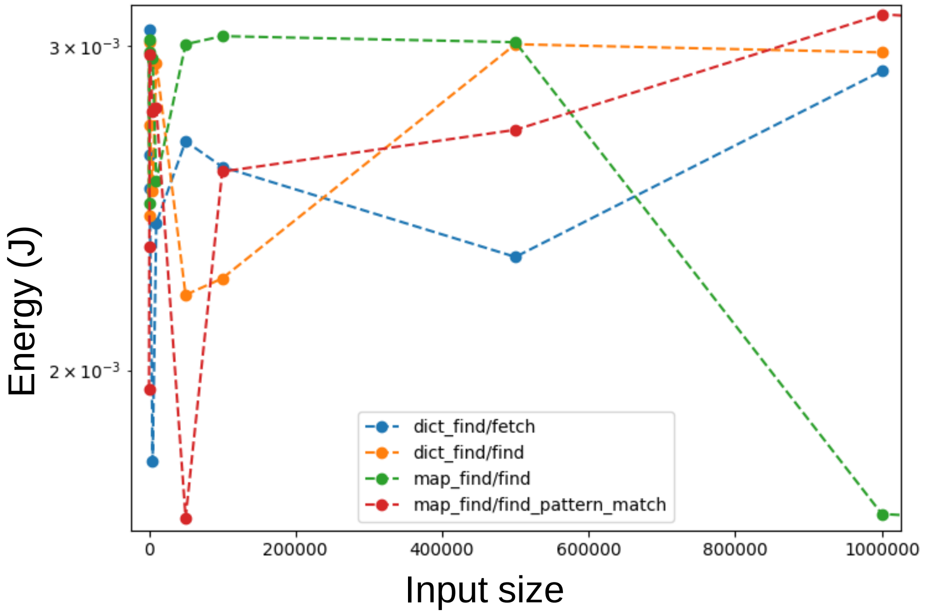
Find - results

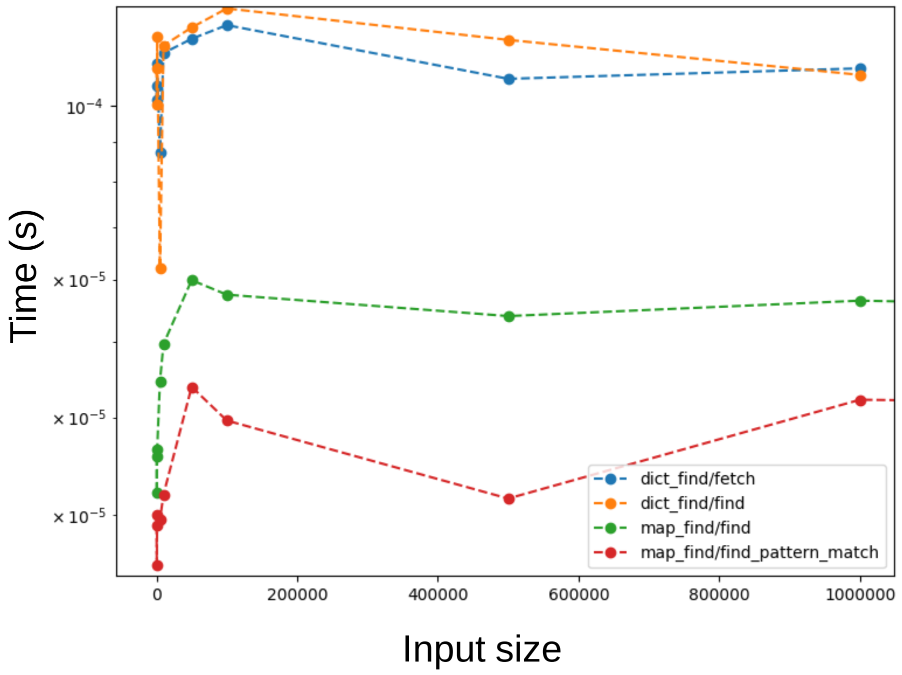




- Finding with pattern matching is faster, but consumes more energy
- Using the find function consumes the least energy but is slower
- Needs more testing

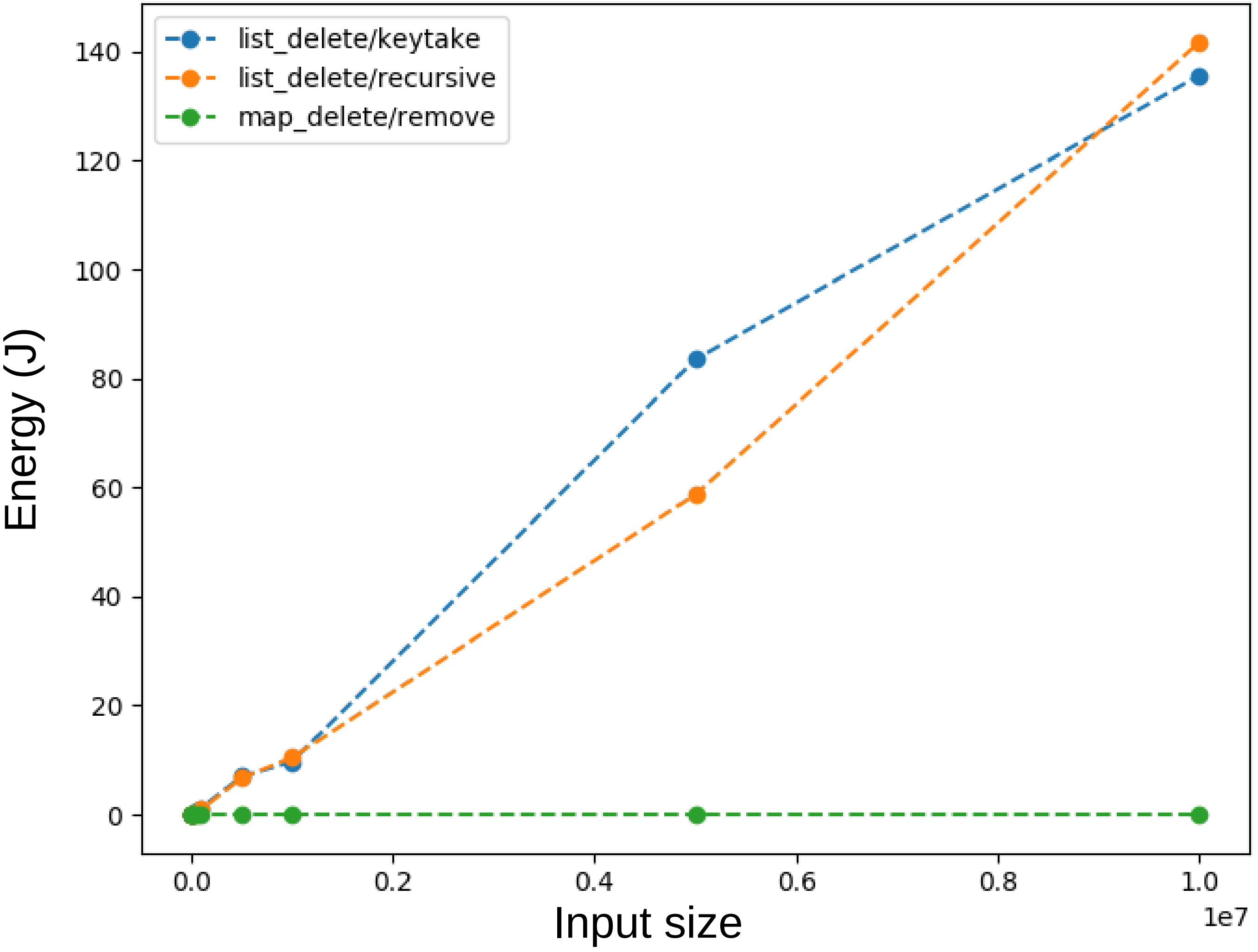
Find - results





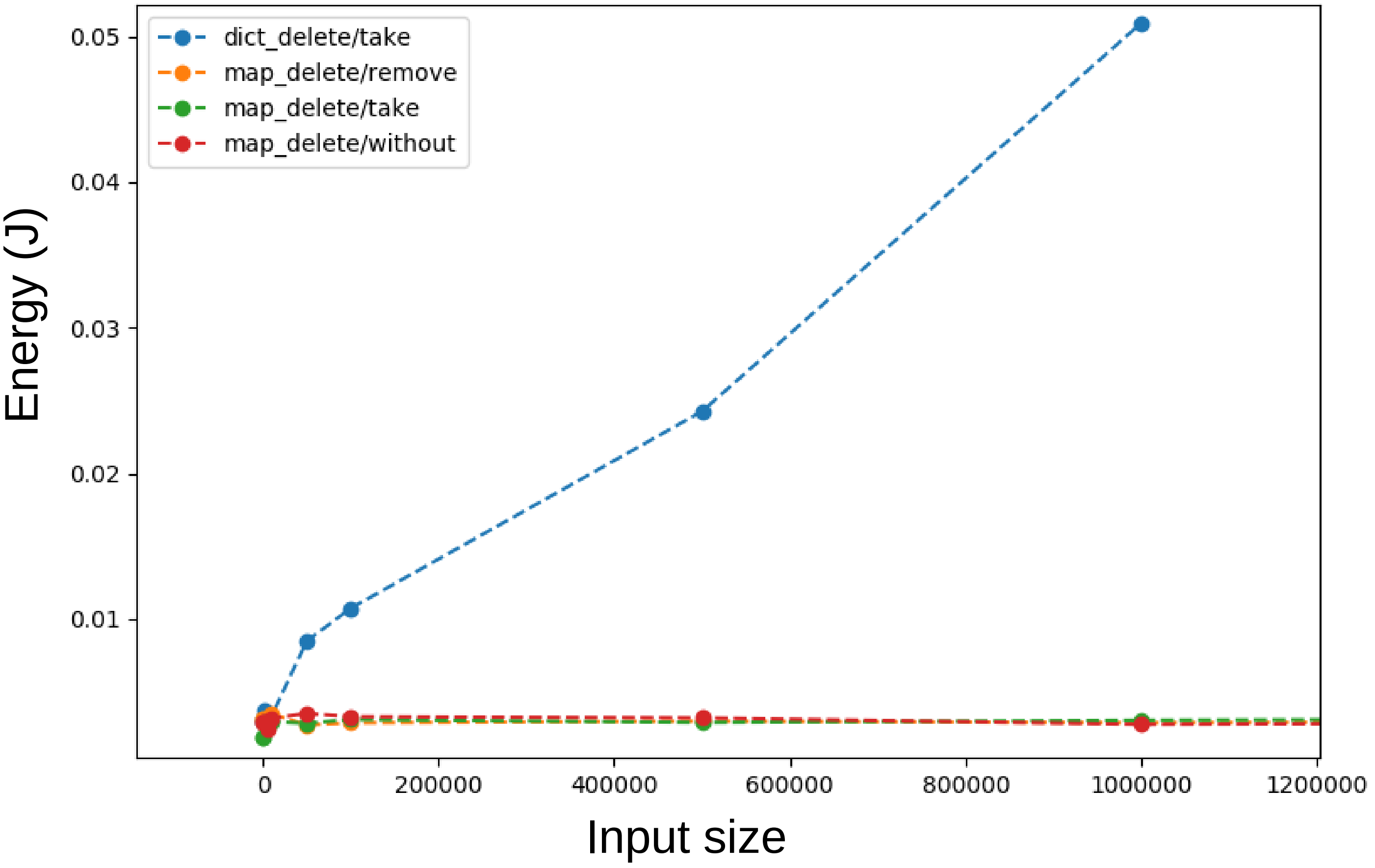
- Finding in a dict is similar to finding in a map,
- Maps are quicker
- Energy consumption may be better for maps (cannot construct large enough dicts)

Delete - results



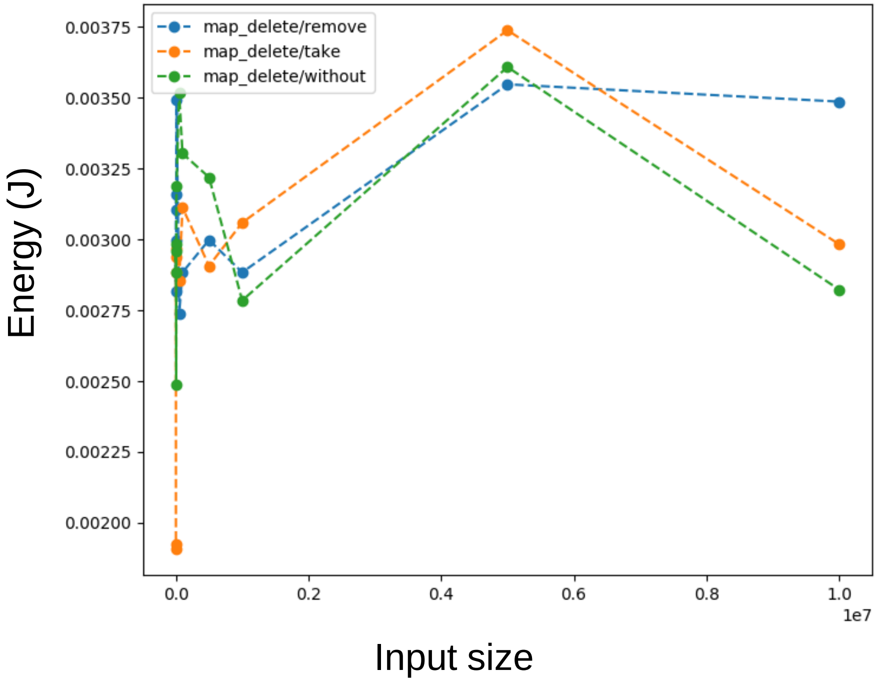
- No conclusive difference between the two ways to delete
- Much worse than deleting from a map (as expected)

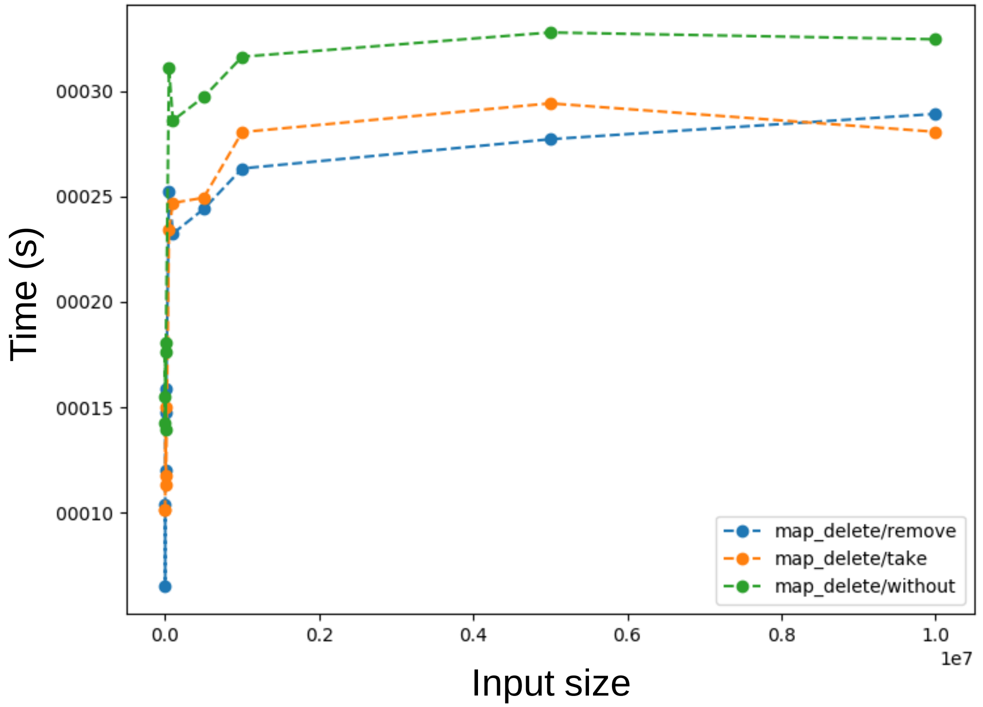
DELETE - RESULTS



- Deleting from a dict is worse than deleting from a map

DELETE - RESULTS





- Deleting with maps:without is the slowest, but consumes the least energy
- Needs more testing

Update - Results

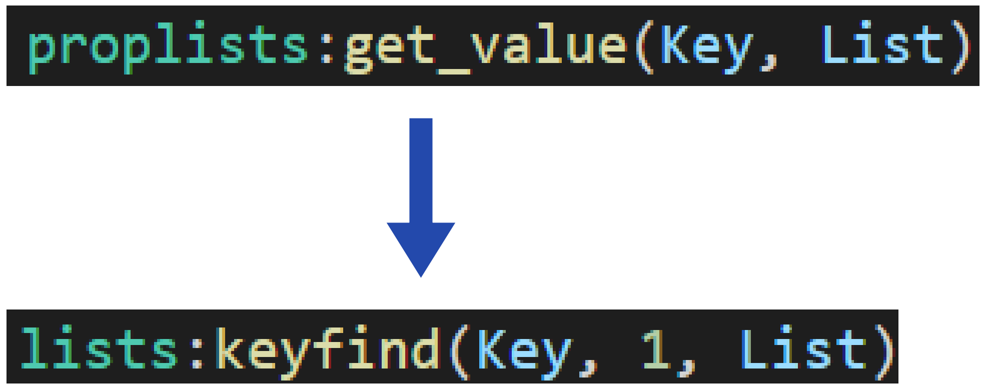
- Results for updating an element in out dictionaries show the same patterns as deleting an element.
- Maps are the most efficient
- Dicts are in the middle
- Lists perform the worst

DATA STRUCTURES - FINDINGS

- Most of the differences between the data structures can be attributed to different representations
- Interesting findings are:
    - We should use lists:keyfind instead of proplists:get_value
    - With maps the quickest solution is not always the more efficient
        - More investigation is needed, because measured values are so small

DATA STRUCTURES - FINDINGS

- Refactoring may be possible between different implementations of functions on the same data structure
- Based on the cost to build every data structure we may be able to decide how many of each operation is needed until it is worth building a different data structure



```
proplists:get_value(Key, List)
```

```
lists:keyfind(Key, 1, List)
```

Conclusions

- Tool for measuring energy consumption
- Higher order functions have an overhead
  - It may be eliminated by rewriting the calls to higher order functions with recursion or list comprehensions
  - Function calls should also be eliminated
- Erlang built-in functions on the same data structures can have significant differences in performance

Future work

- Investigating the cost of spawning processes and sending messages
- Measuring different algorithmic skeletons for parallelization
- Creating refactorings based on our findings using RefactorErl

THANK YOU FOR YOUR ATTENTION!

EFOP-3.6.2-16-2017-00013

innovation informatics infocommunications
ELTE PPKE BME
European Union
SZÉCHENYI 2020
HUNGARIAN GOVERNMENT
European Union European Social Fund
INVESTING IN YOUR FUTURE