

# Evaluation of the impact on energy consumption of lazy versus strict evaluation of data-structures

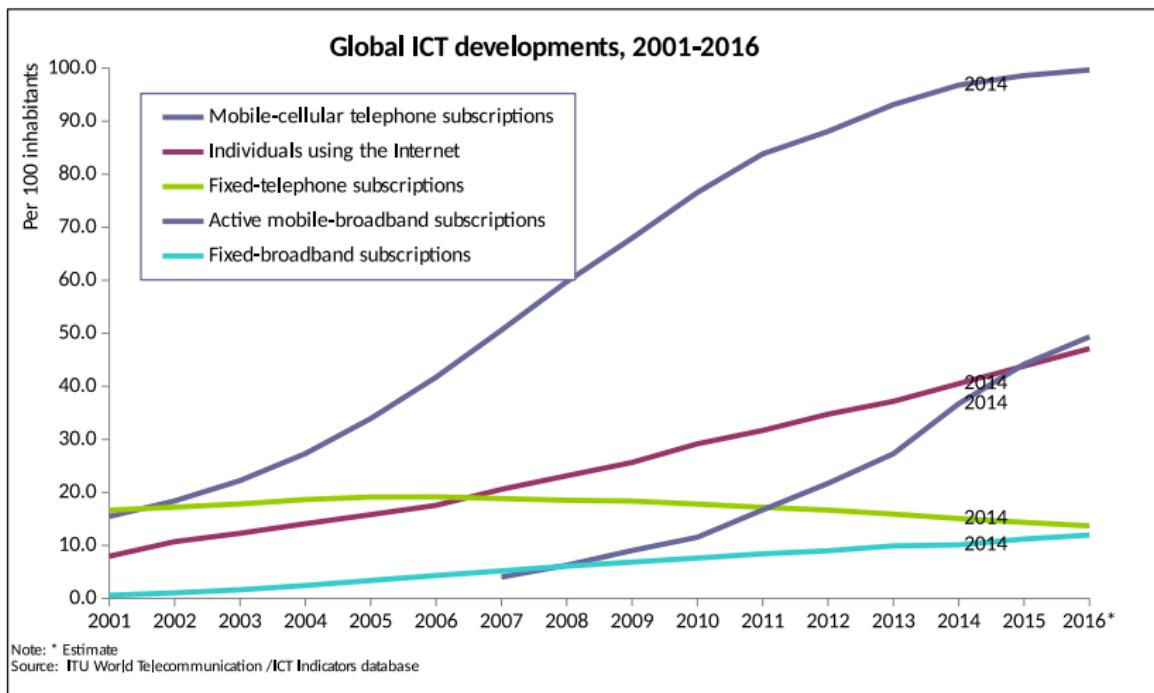
Gilberto Melfe, Alcides Fonseca, João Paulo Fernandes

August 29, 2018

# Introduction

# Introduction

Where are we? Where are we heading?



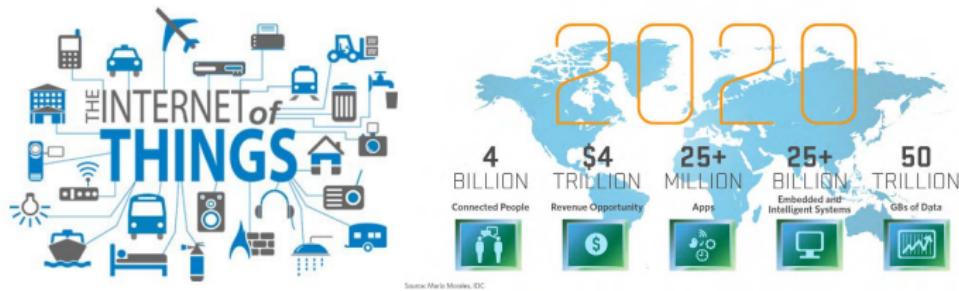
# Introduction

Where are we? Where are we heading?



# Introduction

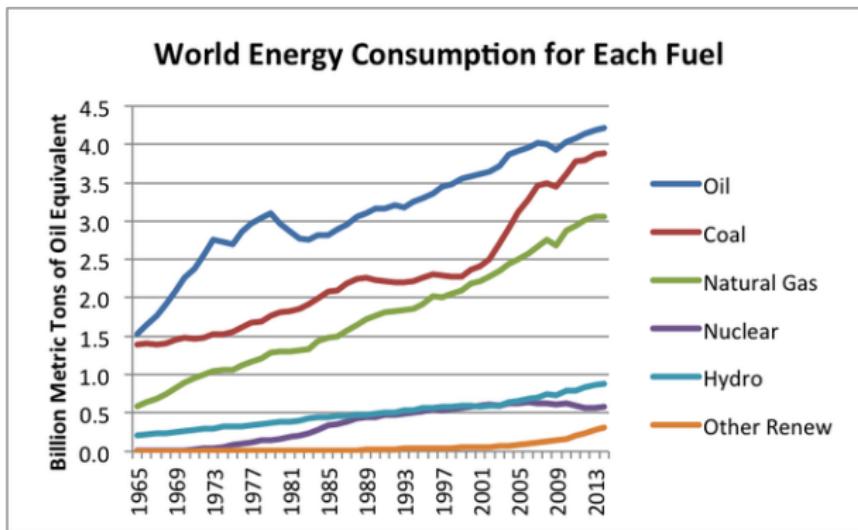
Where are we? Where are we heading?



Source: <http://cdn2.business2community.com/wp-content/uploads/2016/06/internet-of-things.jpg>

# Introduction

And the problem is...



Our Finite World blog by Gail Tverberg is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

# Introduction

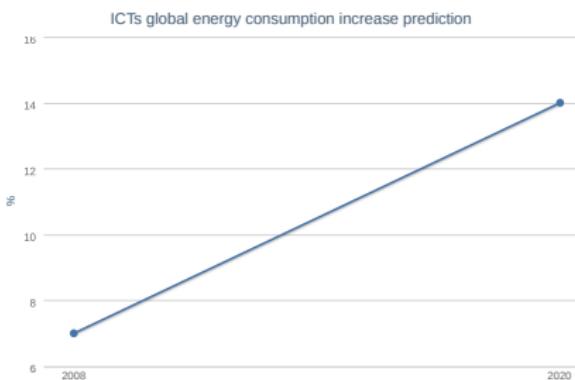
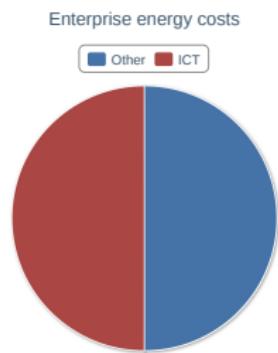
And the problem is...



DIBYANGSHU SARKAR/AFP/Getty Images

# Introduction

Some numbers...



# Introduction

We must...



# Introduction

We intend to...

- Search possible gains in the energy consumption induced by software
  - Maybe between 30% and 90%?
- Focus on the Haskell language

# Introduction

We intend to...

- Search possible gains in the energy consumption induced by software
  - Maybe between 30% and 90%?
- Focus on the Haskell language

# Introduction

We have...

- Investigated the Edison library
  - Refactoring, to use alternative data structure implementations
  - Concerning Package and RAM energy consumptions
  - Compiler optimizations effect
- <http://green-haskell.github.io>

# Introduction

Research Question!

**RQ.** How do execution time and Package and RAM energy consumptions are affected by the use of a lazy or strict data structure implementation?

# Approach

# Data structures

## Maps

`Data.Map` a general-purpose implementation of ordered maps (dictionaries), based on balanced binary trees

`Data.IntMap` an efficient implementation of maps, with integer keys, based on big-endian patricia trees

`Data.HashMap` an implementation of unordered maps from hashable keys to values, optimized for performance, based on hash array mapped tries

# Benchmark

## Benchmark Operations

<i>iters</i>	<i>operation</i>	<i>base</i>	<i>elems</i>
1	add	100000	100000
1000	addAll	100000	1000
1	clear	100000	n.a.
1000	contains	100000	1
5000	containsAll	100000	1000
1	iterator	100000	n.a.
10000	remove	100000	1
10	removeAll	100000	1000
10	retainAll	100000	1000
5000	toArray	100000	n.a.

$$\textit{iters} * \textit{operation}(\textit{base}, \textit{elems})$$

# Benchmark

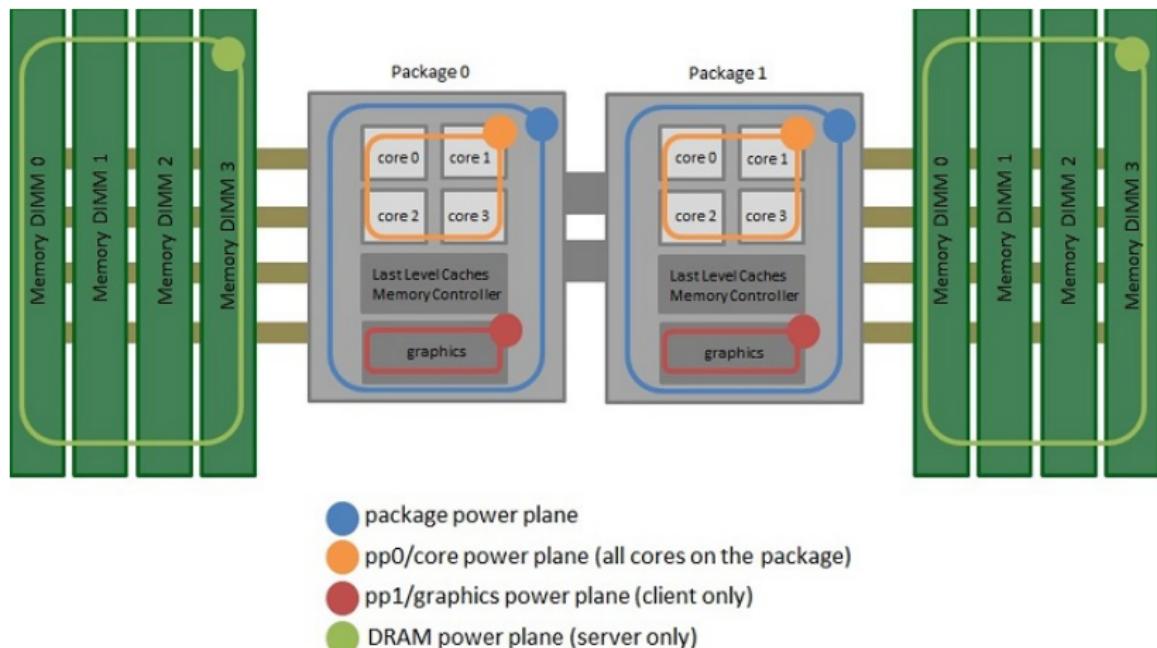
## Benchmark Operations

```
removeAll :: Map Key Datum -> Map Key Datum -> Map Key Datum  
removeAll = difference
```

```
retainAll :: Map Key Datum -> Map Key Datum -> Map Key Datum  
retainAll = intersectionWith const
```

# An interface for measuring energy consumption

RAPL



Source: <https://software.intel.com/en-us/articles/intel-power-governor>

# Benchmark execution and analysis

## Criterion

```
import Criterion.Main

-- Our benchmark harness.
main = defaultMain [
    bgroup "Map/" [
        env (...) -> bench addAllOpDesc $
            nf ( addAllNTimes a b ) addAllNRepeats
    ]
]
```

# Benchmark execution and analysis

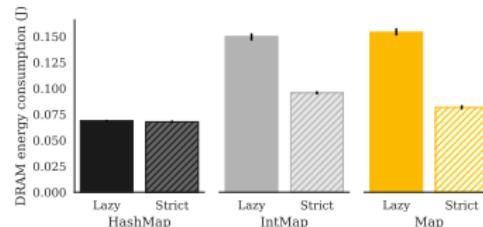
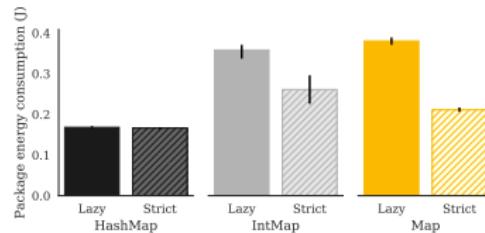
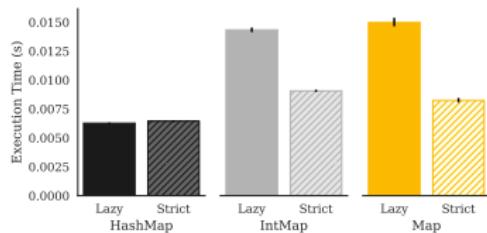
## Criterion

```
benchmarking (...) / addAll_1000_times_1000_elements_to_100000
  time           1.245 s  (1.234 s .. 1.261 s)
                  1.000 R^2  (1.000 R^2 .. 1.000 R^2)
  mean           1.248 s  (1.246 s .. 1.251 s)
  std dev        3.883 ms  (0.0 s .. 4.130 ms)
  packageEnergy:
    iters         31.288   (29.594 .. 33.275)
    y             0.545   (-2.154 .. 5.209)
  dramEnergy:
    iters         10.404   (10.214 .. 10.680)
    y             0.121   (-0.262 .. 0.671)
```

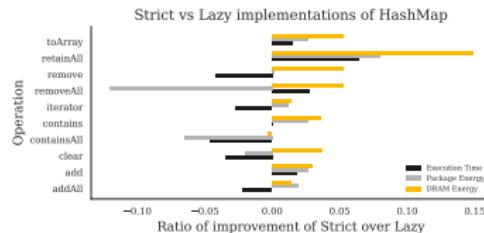
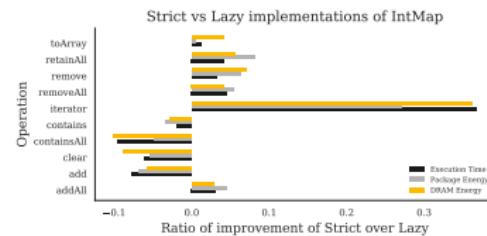
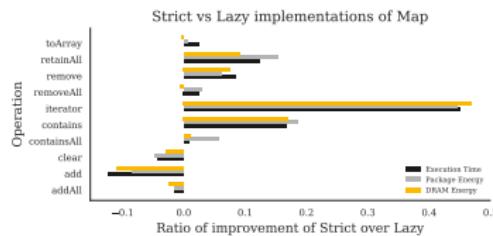
# Results

# Results

## Lazy vs Strict, Iterator



# Results



# Conclusions

# Conclusions

The answer to the RQ. is...

**RQ.** How do execution time and Package and RAM energy consumptions are affected by the use of a lazy or strict data structure implementation?

- Energy consumption is proportional to the execution time, and
- strict evaluation tends to be more efficient in terms of execution time and energy consumption than the lazy evaluation

# Future Work

# Future Work

What's next?

- create/use more real-world benchmarks
- explore laziness vs strictness in more complex algorithms that rely on data structures (sorting or graph search algorithms)

# Evaluation of the impact on energy consumption of lazy versus strict evaluation of data-structures

Gilberto Melfe, Alcides Fonseca, João Paulo Fernandes

August 29, 2018

I'm doing my part



to save energy!

