
Racket-Spark-Light

— Yifan Xing, Alex Cherry —

What does it do?

- What:
 - Transform and analyze large datasets
 - Evaluate and optimize operations used on the datasets
- How:
 - Separate operations into Transformations and Actions
 - Queue up Transformations
 - Combine them to traverse the data as few times as possible

Transformations & Actions

Transformation:

- DataShell -> DataShell
- Evaluated lazily
- Optimized, merged, computed when Action is applied
- E.g:
 - ds-map(func)
 - ds-filter(pred)
 - ds-flatmap(func)

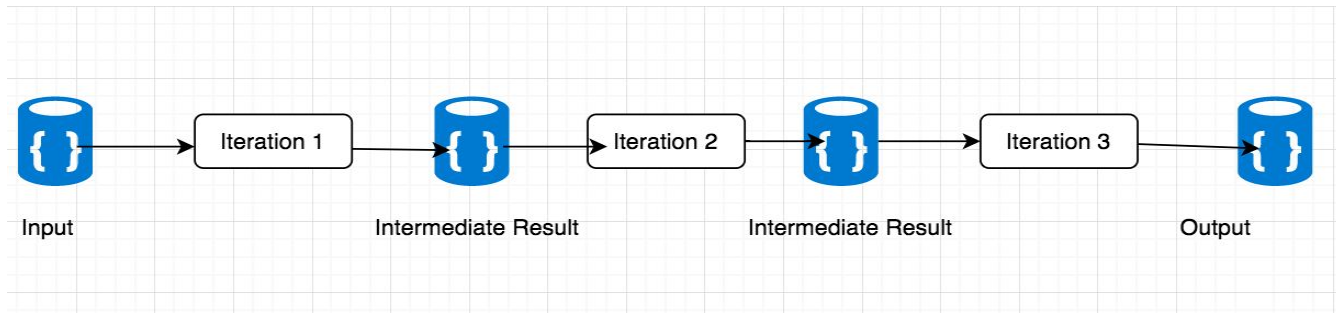
Action:

- DataShell -> Any
- Evaluated eagerly
- Triggers chained transformations and return a result immediately
- E.g:
 - ds-reduce(func)
 - ds-collect()
 - ds-count()

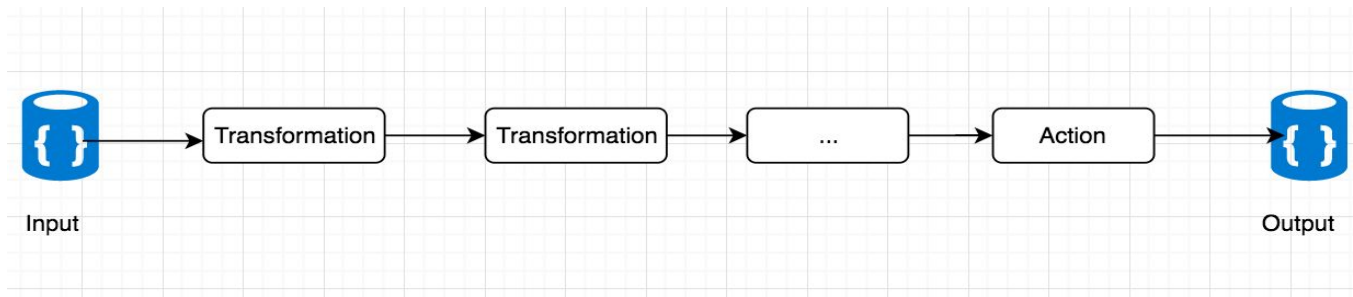
Why should I use it?

- Reduce iterations

Regular
MapReduce



RSL

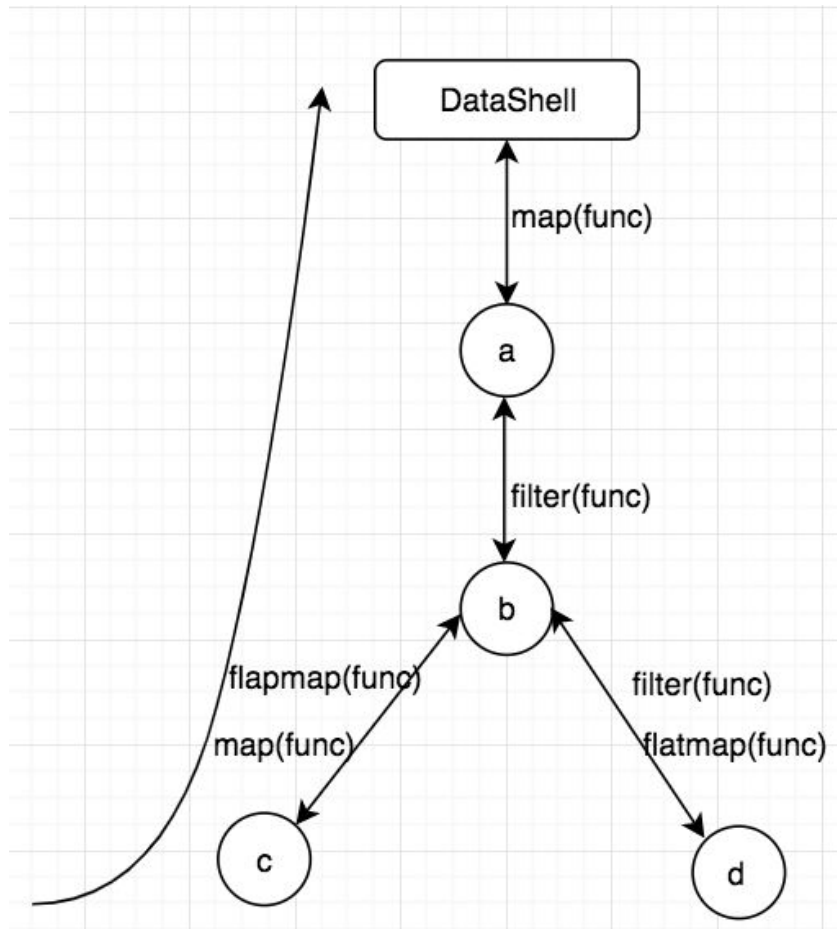


- Stop when sufficient (i.e: `filter(containsErrorStr) -> take(10)`)

Reusable Computations

- Stores DataShells and Transformations
- Reuse previous results
- Fault-tolerant resiliency

```
(save-ds ds (mk-datashell (list 1 2 3 4)))  
(save-ds a (ds-map func ds))  
(save-ds b (ds-filter func a))  
;; get a result  
(define result (ds-reduce func b))  
  
(save-ds c (ds-map func (ds-flatmap func b)))  
(save-ds d (ds-flatmap func (ds-filter func b)))
```



Vocabulary, Grammar, and Scoping Rules

<https://github.com/xingyif/racket-spark-light/tree/master/design>