深入浅出 Transducers

刘家财
http://liujiacai.net/

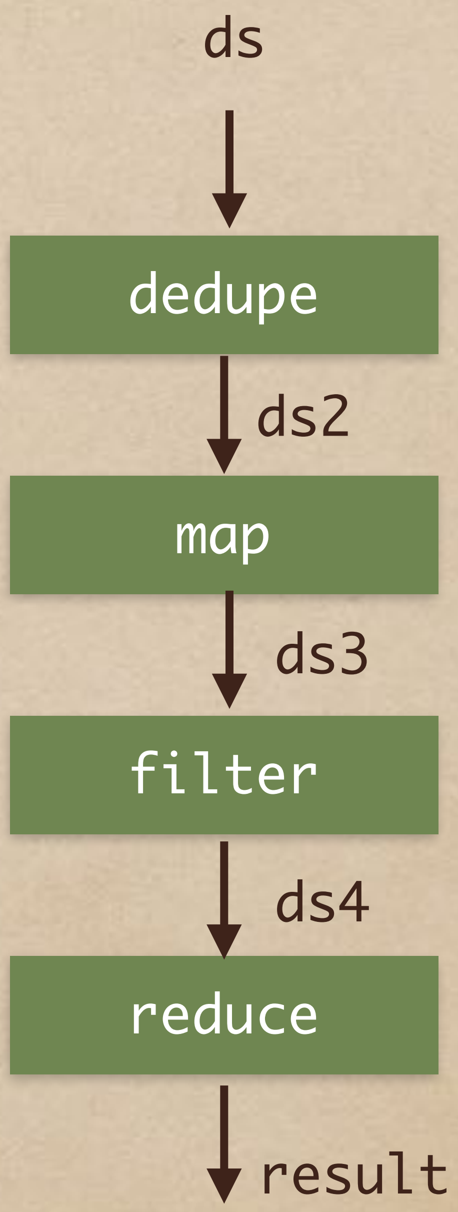
大纲

- Why
- What
  - reducing function
- When
  - 性能
  - 复用
- 使用案例
  - 启动transducer: into/sequence/transducer/eduction
  - 状态
  - early termination (take, drop-while)
- 自己编写

Why

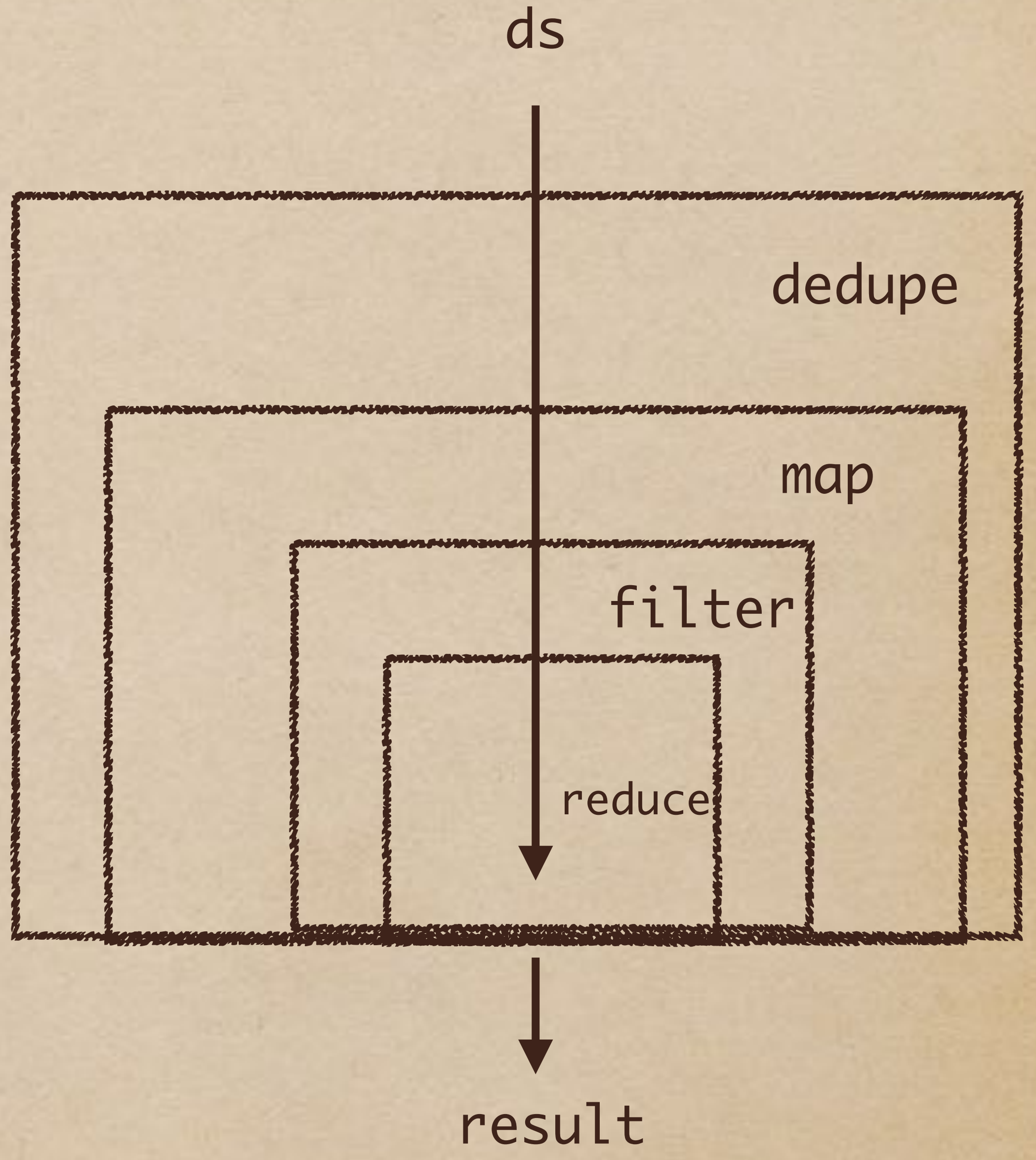
```
(def dataset
  (vec (interleave
           (range 10000)
           (range))))
```

```
(defn workflow [ds]
  (->> ds
       (dedupe)
       (map #(* % %))
       (filter #(= 0 (rem % 111)))
       (reduce +)))

(workflow dataset)
```



Why

```
(def workflow2
  (comp
   (dedupe)
   (map #(* % %))
   (filter #(= 0 (rem % 111)))))


(transduce workflow2 + dataset)
```



Why

- 输入源不是 collection 时，如何复用转化函数?

Transducer 是什么

- Transducer 是一个函数
  - 参数为 reducing function
  - 返回值为 reducing function

Reducing Function

- (accumulate^{N-1}, item) —-> accumulate^{N}
- (conj [1 2] 3) ===> [1 2 3]

Transducer 是什么

```
(fn [xf]
  (fn ([] ...)
      ([result] ...)
      ([result input] ...)))
```

Reduce/fold

一类访问递归类型数据结构的函数

为什么需要 Transducer

- 性能
  - no interim collections
  - no extra boxes
- 复用
  - collection/channel/Observable/Stream

性能比较

```
(defonce dataset (vec (interleave (range 10000) (range))))
```

```
(defn workflow [ds]
  (->> ds
       (dedupe)
       (map #(* % %))
       (filter #(= 0 (rem % 111)))
       (reduce +)))

(bench (workflow dataset))
```

Execution time mean : 1.794328 ms

```
(def workflow2
  (comp
   (dedupe)
   (map #(* % %))
   (filter #(= 0 (rem % 111)))))

(bench (transduce workflow2 + dataset))
```

Execution time mean : 1.143402ms

复用

```clojure
(require '[clojure.core.async :refer [>! <! <!!] :as a])

(def xform (comp (filter odd?) (map inc)))

(defn process [items]
  (let [out (a/chan 1 xform)
        in (a/to-chan items)]
    (a/go-loop []
      (if-some [item (<! in)]
        (do
          (>! out item)
          (recur))
        (a/close! out)))
    (<!! (a/reduce conj [] out))))

(process (range 10))
;; [2 4 6 8 10]
```

使用案例

- 1.7 版本，以下函数被重写，输入一个参数时返回一个 transducer
    - `map cat mapcat filter remove take take-while take-nth drop drop-while replace partition-by partition-all keep keep-indexed map-indexed distinct interpose dedupe random-sample…`

启动 transducer

```clojure
(def nums (range 20))

(def xf (comp (filter even?)
              (map inc)))


(transduce xf conj nums)
(into [] xf nums)
;; [1 3 5 7 9 11 13 15 17 19]

(sequence xf nums)
(eduction xf nums)
;; (1 3 5 7 9 11 13 15 17 19)
```

transduce/into/sequence/eduction

- transduce 与 reduce 类型， 非惰性
- into 内部使用 transduce 实现
- sequence，惰性，cache 结果
- eduction，惰性，没有 cache 结果，每次计算

transduce/into/sequence/eduction

```
(def cnt (atom 0))
(take 10 (transduce (map #(do (swap! cnt inc) %)) conj () (range 1000)))
;; (999 998 997 996 995 994 993 992 991 990)
@cnt
;; 1000


(def cnt1 (atom 0))
(let [res (eduction (map #(do (swap! cnt1 inc) %)) (range 10))]
  (conj (rest res) (first res))
  @cnt1)
;; 20

(def cnt2 (atom 0))
(let [res (sequence (map #(do (swap! cnt2 inc) %)) (range 10))]
  (conj (rest res) (first res)) ; (2)
  @cnt2)
;; 10
```

状态

```clojure
(defn dedupe []
  (fn [xf]
    (let [prev (volatile! ::none)]
      (fn
        ([] (xf))
        ([result] (xf result))
        ([result input]
          (let [prior @prev]
            (vreset! prev input)
              (if (= prior input)
                result
                (xf result input))))))))
```

Early termination

- reduced 返回一个reduced值，表明reduction结束
- reduced?
- deref/@ 取出 reduced 包含的值

Early termination

```
(defn take [n]
  (fn [rf]
    (let [nv (volatile! n)]
      (fn
        ([] (rf))
        ([result] (rf result))
        ([result input]
         (let [n @nv
               nn (vswap! nv dec)
               result (if (pos? n)
                        (rf result input)
                        result)]
           (if (not (pos? nn))
             (ensure-reduced result)
             result)))))))
```

自己编写transducer

```
(defn template-transducer [xf]
  (fn
    ;; SET-UP
    ([]                 (xf))
    ;; TEAR-DOWN
    ([result]           (xf result))
    ;; PROCESS
    ([result input]     (xf result input))))
```

扩展

- https://labs.uswitch.com/transducers-from-the-ground-up-the-essence/
- use transducer if possible

Thank You.

群名称：SICP读书群

群　号：119845407

公众号