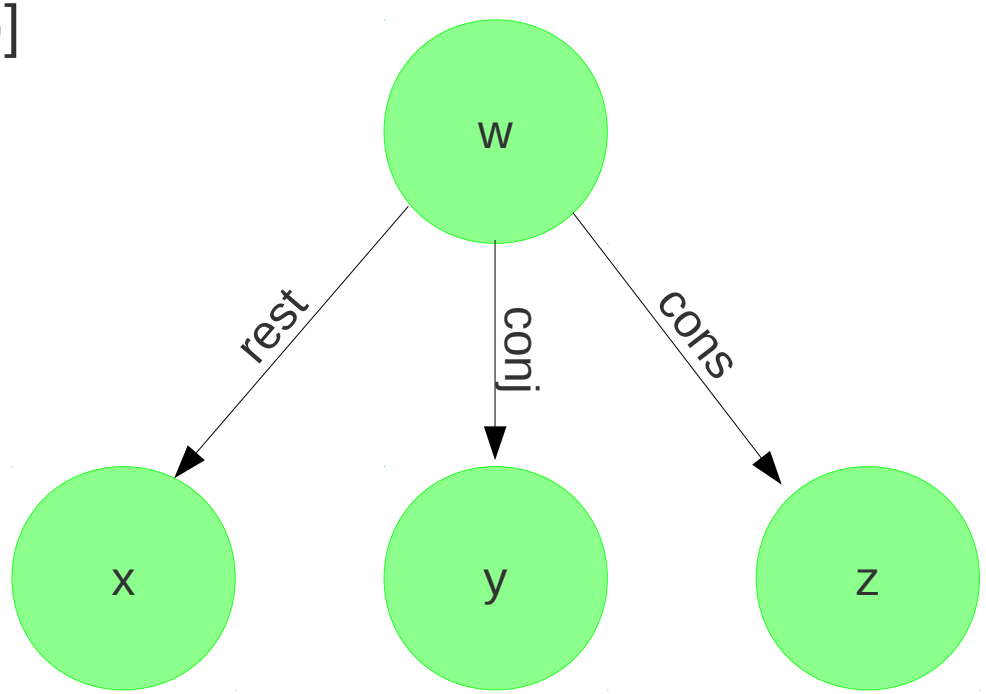
Clojure Collections

Steven Reynolds

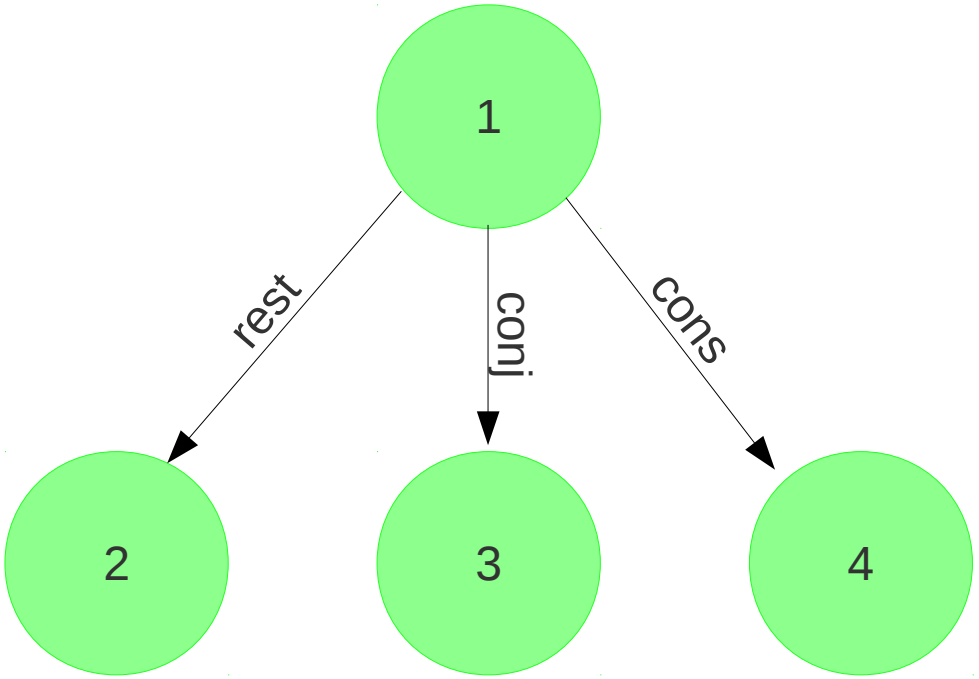
Slides will be available at www.slreynolds.net

Functional Data Structures don't change

```
(let [w '(2 3 4)
      x (rest w)
      y (conj  w 1)
      z (cons 1 w)]
    ...
```

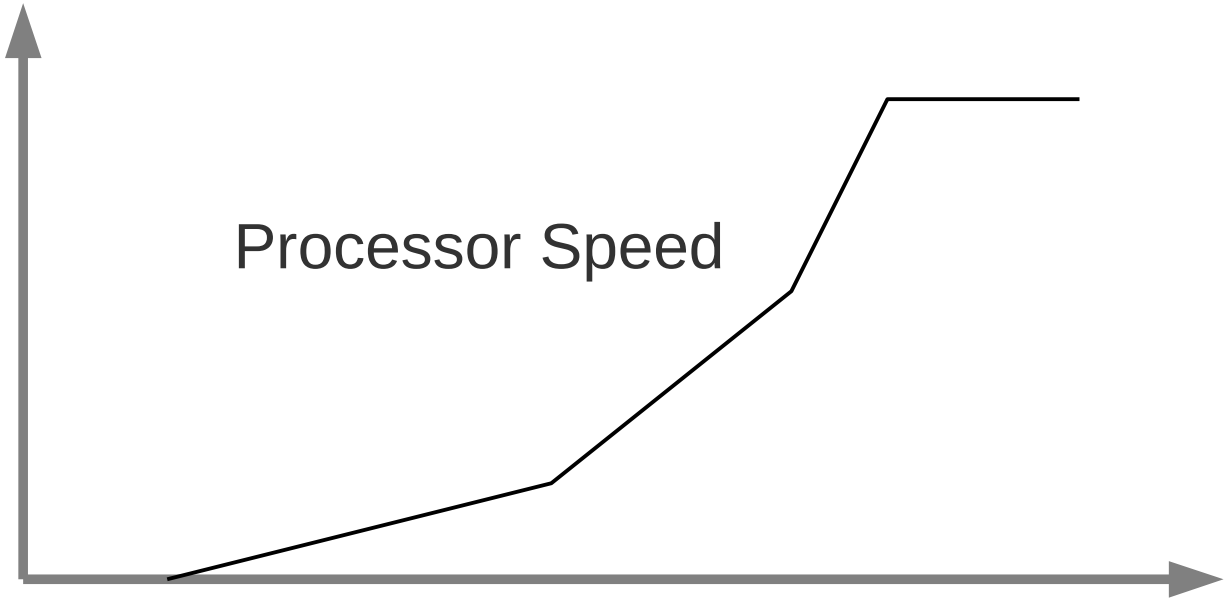


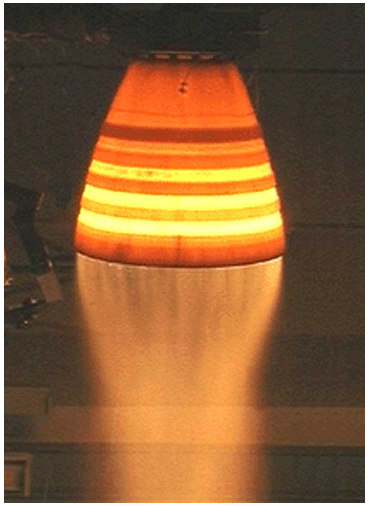
Multiple versions of
the same data structure

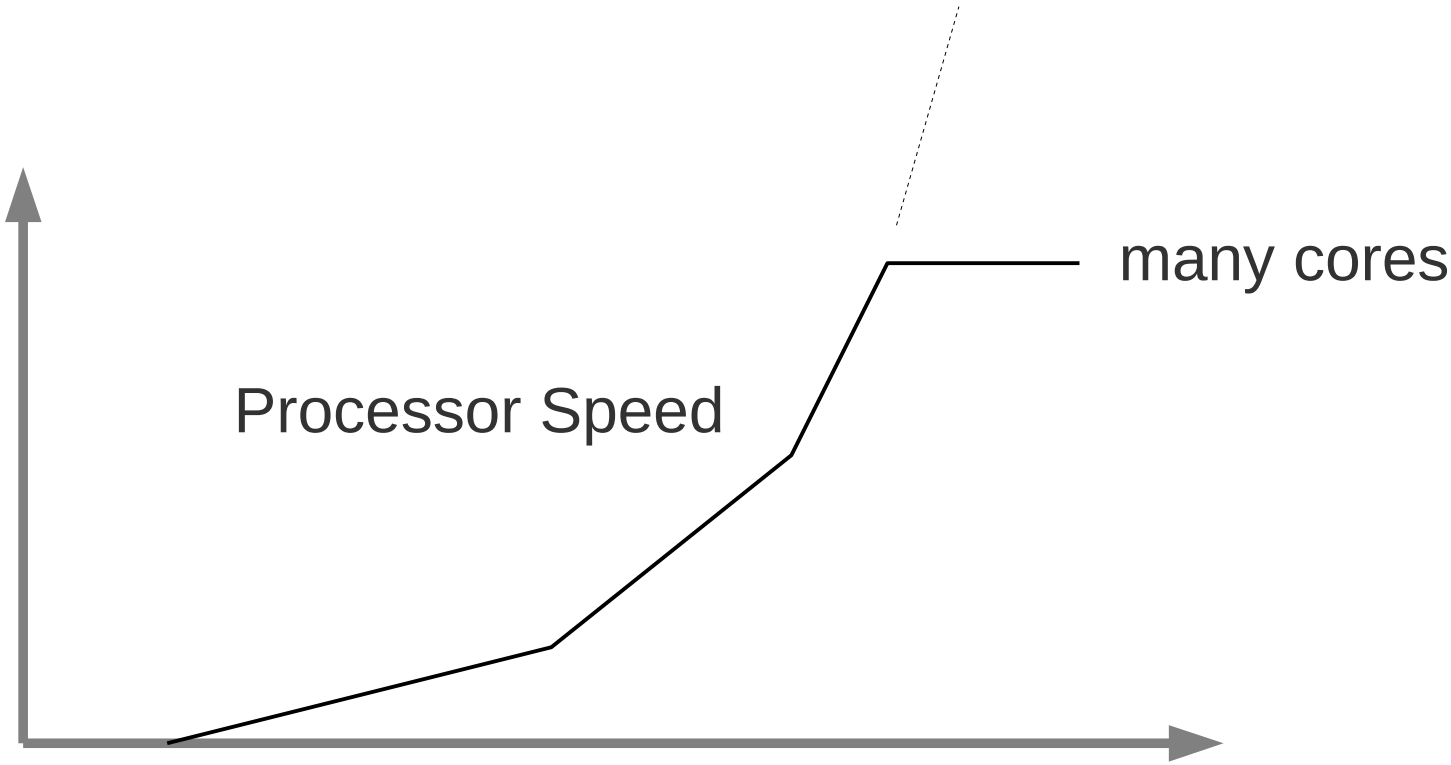


why?

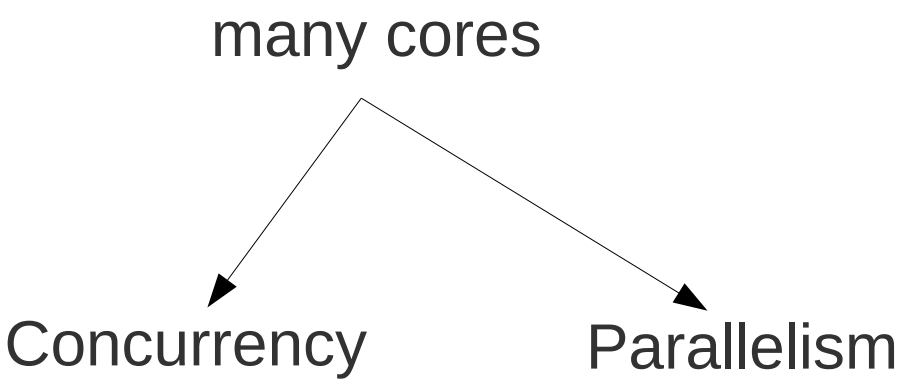
why?






many cores
Processor Speed

Herb Sutter “The Free Lunch is Over” (2005)



Lions, Tigers and Bears. Oh my.
- Dorthy

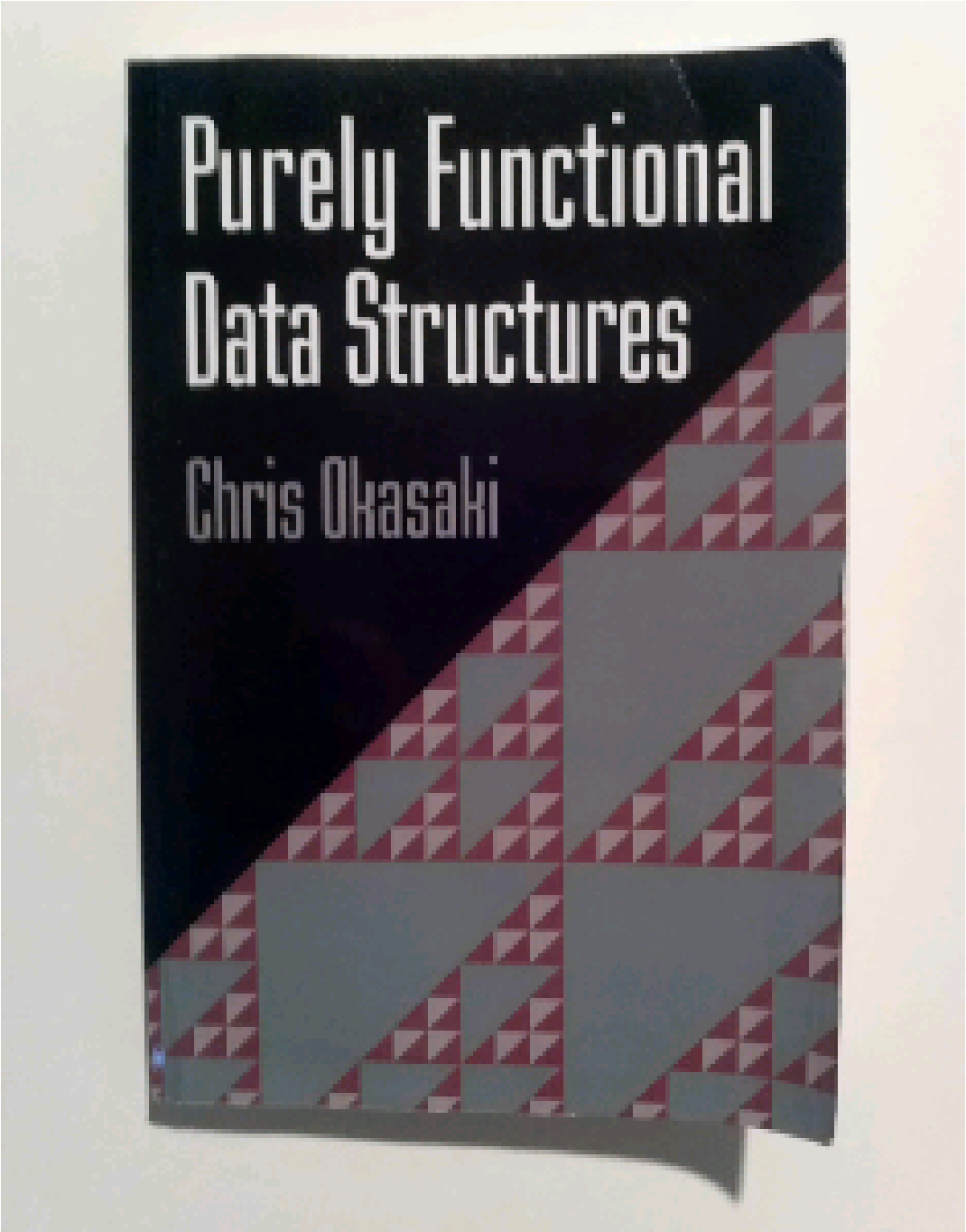
many cores

Concurrency

Parallelism

Functional Data Structures can help:

They don't change


Purely Functional
Data Structures
Chris Okasaki

... functional programming's stricture against destructive updates (i.e. assignments) is a staggering handicap, tantamount to confiscating a master chef's knives.

- Chris Okasaki

```
String brother = "brother";
String the = brother.substring(3, 6);

saver.save(new Object[]{brother,the},
           new String[]{"brother","the"},options);
```

```java
private static class Foo {
   private int count;
   private String name;
   private double level;
   public Foo(int count, String name, double level) {
      super();
      this.count = count;
      this.name = name;
      this.level = level;
   }
}

private static class Bar {
   private Foo foolet;
   private int ID;
   public Bar(Foo foolet, int iD) {
      super();
      this.foolet = foolet;
      ID = iD;
   }
}
```

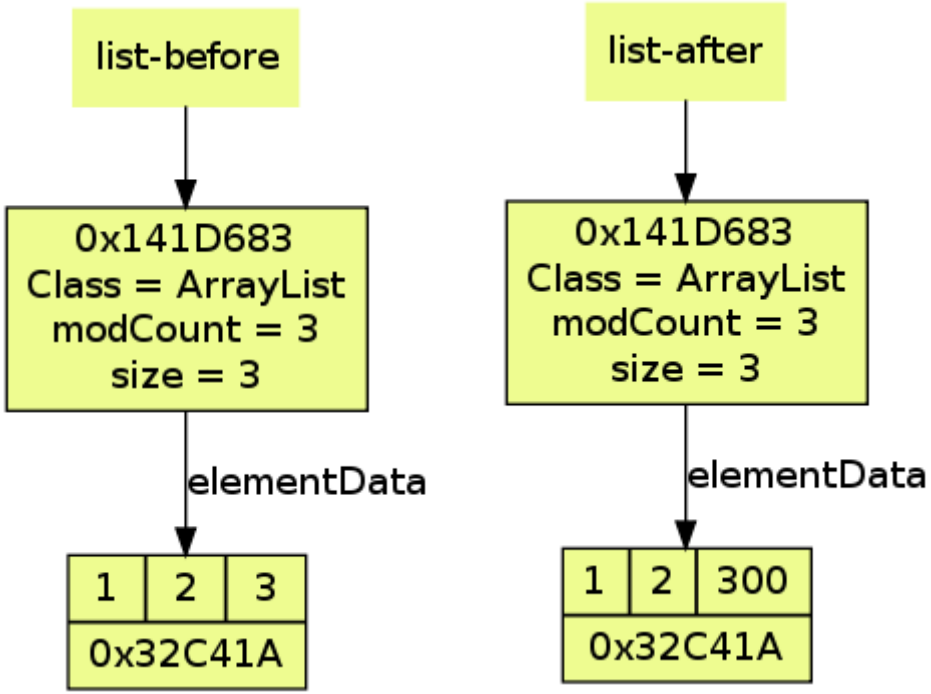
```java
Foo foo = new Foo(5,"bob", 7.0f);
Bar bar = new Bar(foo, 1234567);

saver.save(new Object[]{foo,bar},
        new String[]{"foo","bar"},options);
```

```java
ArrayList<Integer> list = new ArrayList<Integer>(3);
list.add(1);
list.add(2);
list.add(3);
saver.save(list, "list", "alist.dot");
```

```
ArrayList<Integer> list2 = new ArrayList<Integer>(3);
list2.add(1);
list2.add(2);
list2.add(3);
saver.save(new Object[]{list2},
            new String[]{"list-before"},options);

list2.set(2,300);
saver.save(new Object[]{list2},
         new String[]{"list-after"},options);
```



Strings will now be in-lined

List

```
(defn list-ex []
  (let [w (list 2000 3000 4000)
        x (rest w)
        y (conj  w 1000)
        z (cons 1000 w)
        saver (ReflectiveSaver.)]
    (. saver save (list w x y z) '("w" "(rest w)" "(conj w 1)"
"(cons 1 w)") {ExporterOptions/OUTPUT_PATH
"/home/steven/graphs/list_ex.dot"})
    ))
```

Vector

```
(defn vec-ex []
  (let [w (vector 2 3 4)
        x (pop w)
        y (conj w 1)
        z (assoc w 2 "something")
        saver (ReflectiveSaver.)]
    (. saver save (list w x y z) '("x" "(pop x)" "(conj x 1)"
"(assoc x 2 \"something\")") {ExporterOptions/OUTPUT_PATH
"/home/steven/graphs/vec_ex.dot"})
    ))
```

```
(defn large-vec []
  (let [x (vec (take 35 words))
        y (assoc x 34 "something")
        saver (ReflectiveSaver.)]
    (. saver save (list x y) '("x" "(assoc x 34 \"something\")")
{ExporterOptions/OUTPUT_PATH "/home/steven/graphs/large_vec.dot"})
    ))
```

Map

```
(defn amap-ex []
  (let [x {1 "one" 2 "two"}
        y (assoc  x 3 "three")
        saver (ReflectiveSaver.)]
    (. saver save (list x y) '("x" "after assoc ...")
{ExporterOptions/OUTPUT_PATH "/home/steven/graphs/amap_ex.dot"})
    ))
```

```
(defn hmap-ex []
  (let [x (hash-map 1 "one" 2 "two")
        y (assoc  x 3 "three")
        saver (ReflectiveSaver.)]
    (. saver save (list x y) '("x" "after assoc ...")
{ExporterOptions/OUTPUT_PATH "/home/steven/graphs/hmap_ex.dot"})
    ))
```

```
(defn large-hmap []
  (let [x (apply hash-map (interleave (range 1 20) words))
        y (assoc x  21 "physics")
        saver (ReflectiveSaver.)]
    (. saver save (list x y) '("x" "after assoc")
{ExporterOptions/OUTPUT_PATH
"/home/steven/graphs/large_hmap.dot"})
    ))
```

Lazy

```
(defn simple-range [i limit]
  (lazy-seq
            (when (< i limit)
              (cons i (simple-range (inc i) limit)))))

(defn lazylist1 []
   (let [w (simple-range 1 4)
         saver (ReflectiveSaver.)]
     (. saver save (list w) '("original simple-range")
{ExporterOptions/OUTPUT_PATH
"/home/steven/graphs/lazylist1_ex.dot"})
     ))
```

```
(defn lazylist2 []
   (let [w (simple-range 1 4)
         x (rest w)
         saver (ReflectiveSaver.)]
     (. saver save (list w x) '("original simple-range"
"rest of ...") {ExporterOptions/OUTPUT_PATH
"/home/steven/graphs/lazylist2_ex.dot"})
     ))
```

Warning

Graphs show internal details that may change

Based on clojure 1.2.0

Opinions expressed are due to Houston weather

Your mileage may vary

Smoking is harmful to your health

…

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore

Thank you

Rich Hickey
Clojure Developers
Phil Bagwell
Chris Okasaki