Final words on functional programming

Advantages of functional progs

- Functional programming more concise

  “one line of lisp can replace 20 lines of C”

  (quote from http://www.ddj.com/dept/architect/184414500?pgno=3)

- Recall reverse function in OCaml:

```
let reverse = fold (::) [];;
```

- How many lines in C, C++?

Can better reason about progs

- No side effects. Call a function twice with same params, produces same value

- As a result, computations can be reordered more easily
- They can also be parallelized more easily

Industry

- From the authors of map reduce: "Inspired by similar primitives in LISP and other languages"

  http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0003.html

- The point is this: programmers who only know Java/C/C++ would probably not have come up with this idea

- Many other similar examples in industry

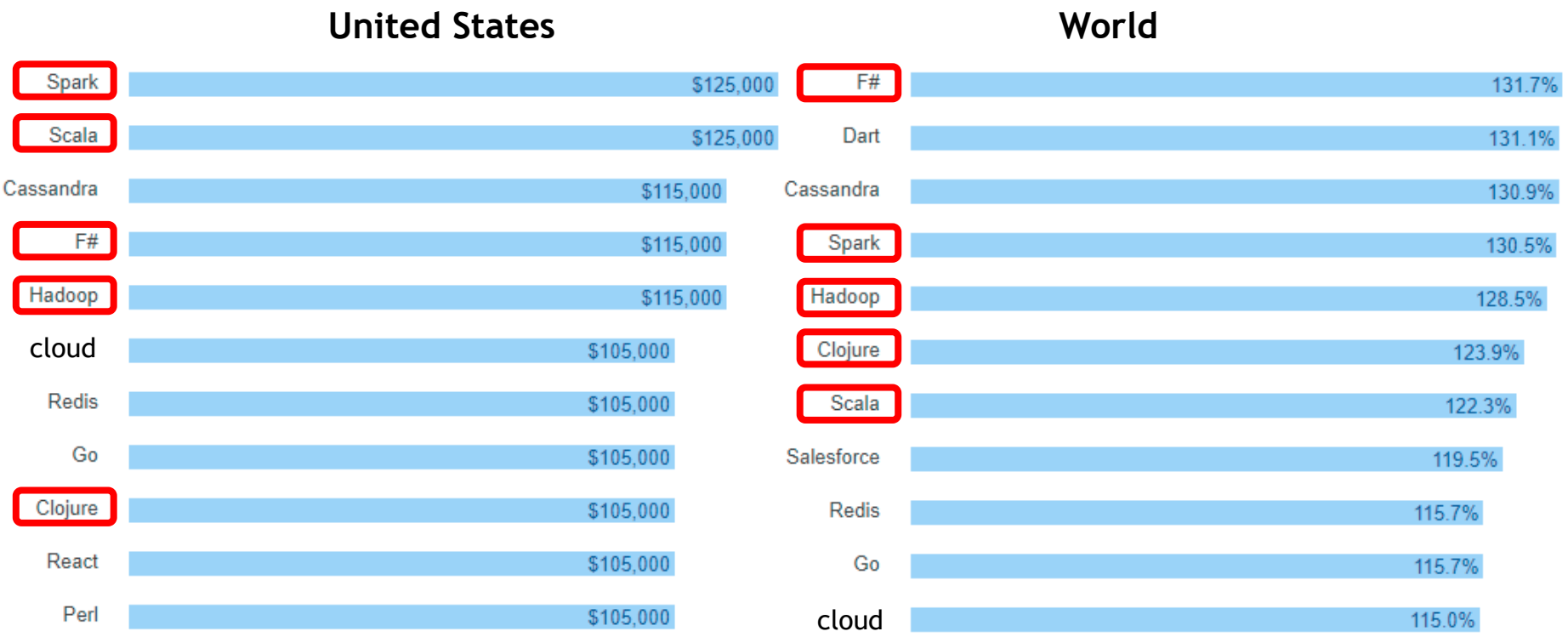
Industry

- Microsoft: F#, inspired by Ocaml
  https://channel9.msdn.com/blogs/pdc2008/tl11
- Jane Street Capital: uses Ocaml for their trading software
- Facebook: Infer program analysis tool implemented in Ocaml
- Facebook: Sigma malware detection tool implemented in Haskell
- Google: map reduce, influenced by FP
- Twitter: uses Scala for their back-end (Scala has roots in FP and OO)

Stack Overflow Survey

Top Paying by Language (self reported)

https://insights.stackoverflow.com/survey/2016#technology-top-paying-tech



: functional or heavily influenced by functional

Final words on Constraint Logic Programming

Different way of thinking

- State constraints, and ask solver to get solution
- Very powerful paradigm: separates *constraint generation* from *constraint solving*
- You generate the constraints, and the used off-the-shelf solver
- You will see a very powerful application of this in the last Python assignment

Industry

- Used in Watson, IBM's Jeopardy-winning computer

- Used in various niche industries, eg solving constraints about manufacturing (in many ways has been superseded in this respect by AI/statistical methods)

Recap of the course so far

- 4 weeks of functional with Ocaml
- 1 week of constraint logic programming with Prolog
- Next: 4 weeks of OO with Python

OCaml/Python comparison

	ML	Python
PL paradigm		
Basic unit		
Types		
DataModel		

OCaml/Python comparison

	ML	Python
PL paradigm	functional	OO/imperative
Basic unit	Expr/value	Objects/ messages
Types	statically	dynamicaclly
DataModel	env lookup	“pointers” to mutable objs

Dynamic vs. Static, OO vs. Func

	Statically typed	Dynamically typed
OO	Java	Python, Smalltalk
Functional	Ocaml, Haskell	Lisp/Scheme

Python

- Python has a very relaxed philosophy
  - if something "can be done" then it is allowed.

- Combination of dynamic types + everything is an object makes for very flexible, very intuitive code.

No static types

- No static type system to "prohibit" operations.
- No more of that OCaml compiler giving you hard-to-decypher error messages!
- And... No need to formally define the type system (although still need to define the dynamic semantics somehow)

Similarities to Ocaml

- Uniform model: everything is an object, including functions
- Can pass functions around just as with objects
- Supports functional programming style with map and fold

Let’s fire it up!

- Ok, let’s give it a try...
- See py file for the rest...