St. Louis Clojure

Powderkeg

Christopher Mark Gore

cgore.com

Tuesday, March 21, AD 2017

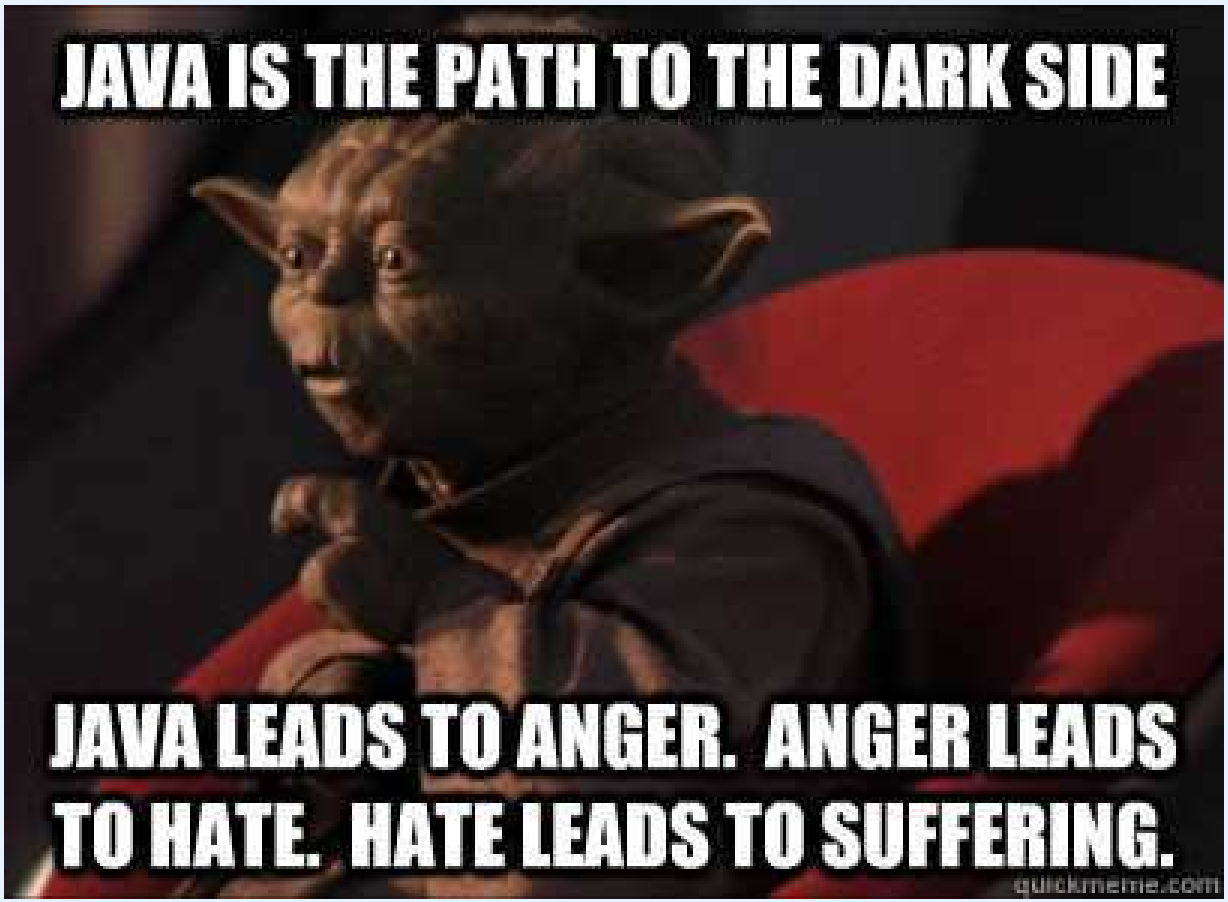
We write Clojure at The Climate Corporation, and we're hiring! Come work with us!

Especially now that Bayer is buying us!

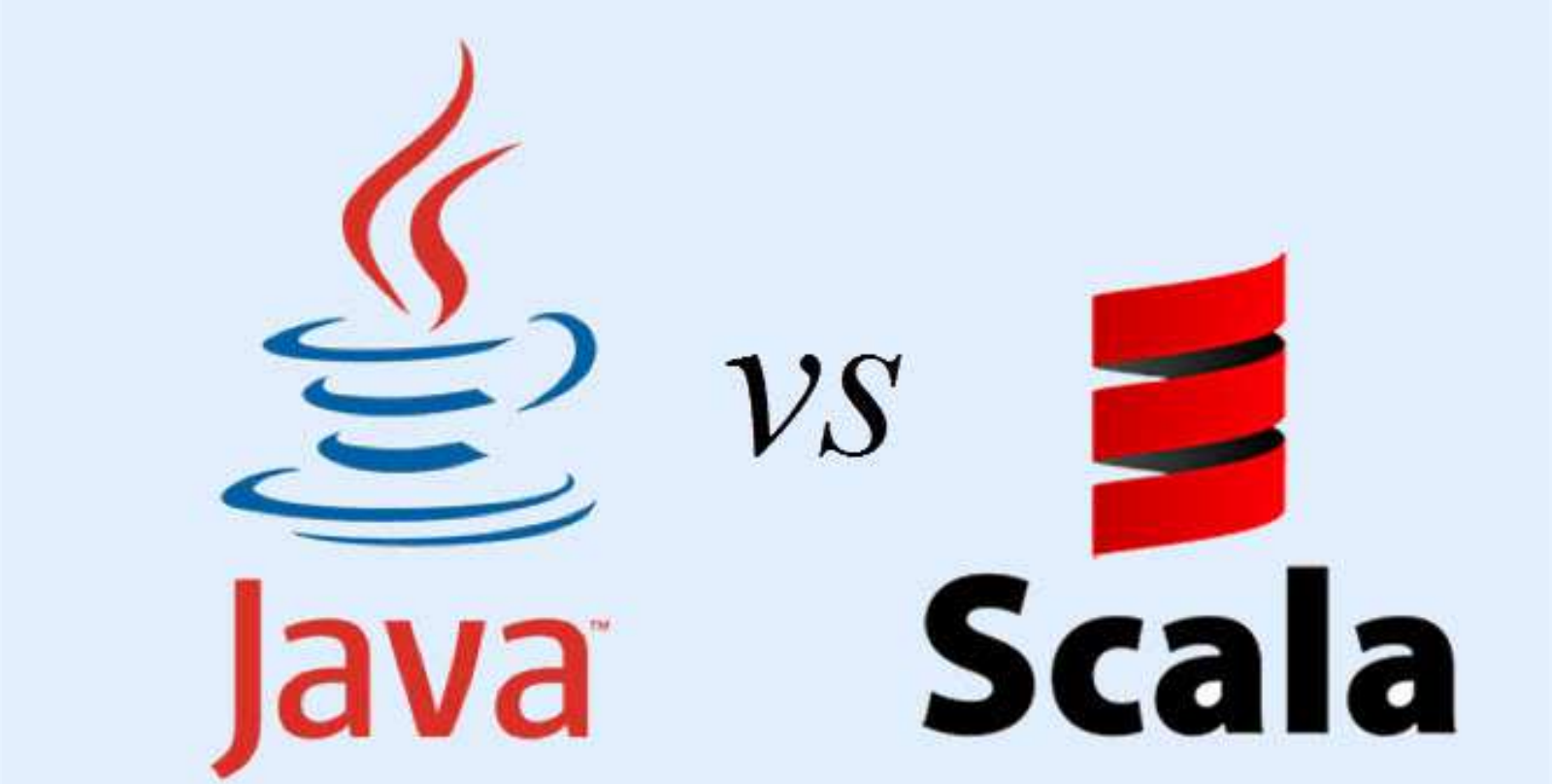
It's a pretty cool place to work, we've even got a giant globe to play with.

Clojure is a lisp.

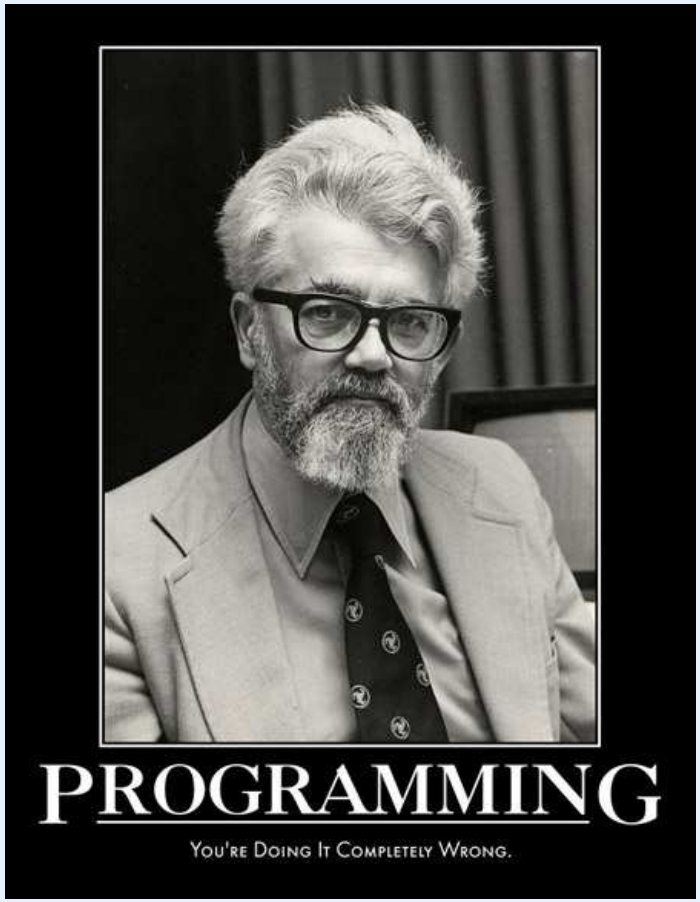
Clojure is Lisp on the JVM.



Scala is also on the JVM.



But Scala isn't a lisp.



Apache Spark is really cool, but it's in Scala.

Apache Spark is an open source cluster computing framework.

- Based on the RDD, resilient distributed dataset.
- An RDD is basically a distributed read-only multiset.
- RDDs allow for the power of MapReduce but with a lot more flexibility.
- RDDs can be treated as shared memory.
- This allows for iterative algorithms, not just map and then reduce operations.

What's a Clojurian to do?

- Use Scala? Nope.
- Here at Climate, we made `clj-spark` well before I came to work here, which was a good start.
- This eventually became Flambo, which is pretty good, but doesn't exactly feel like Clojure.

Let's make another library!

- Igor Ges and Christophe Grand introduced a new library called *Powderkeg* to work with Apache Spark in Clojure.
- It looks like normal Clojure code, thanks to transducers!
- But . . . it's still really early alpha.

It looks almost like normal Clojure.

```
1  ;; 'normal' Clojure
2  (into [] (map #(* % %))
3                [1 2 3 4 5])
4
5  ;; Flambo
6  (-> (f/parallelize sc [1 2 3 4 5])
7      (f/map (f/fn [x] (* x x)))
8      f/collect)
9
10 ;; Powderkeg
11 (into [] (keg/rdd [1 2 3 4 5]
12                   (map #(* % %))))
```

So what exactly do you do with a Spark cluster?

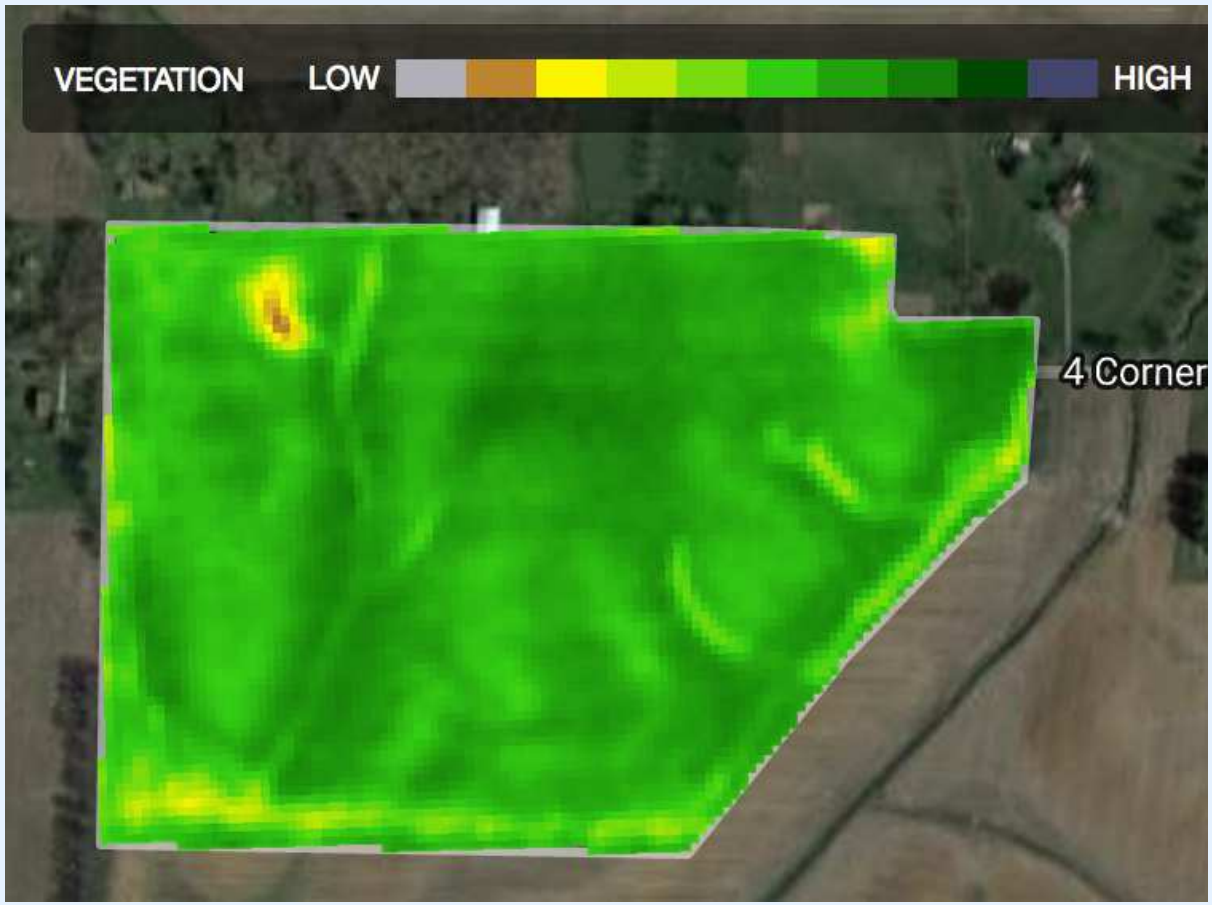
- Make a big list of stuff, and RDD.
- Map on that RDD …
- Filter down that RDD …
- Reduce on that RDD …
- …

Until you have the final result of your computation. But, all this mapping, reducing, and filtering has occurred on multiple machines, not just one machine.

Why we care at Climate (besides it just being cool.)

- Most of our imagery generation is quite amenable to this sort of parallellization.
- We currently have to run on somewhat large (and expensive) instances because we currently operate on a per-field basis.
- The instance type required for a 100-acre field is a lot cheaper than what we need to calculate on a 3,000-acre field.
- But if we can span across multiple instances per-field, then we can use smaller and cheaper instances, just more of them.

Sample Vegetation Map



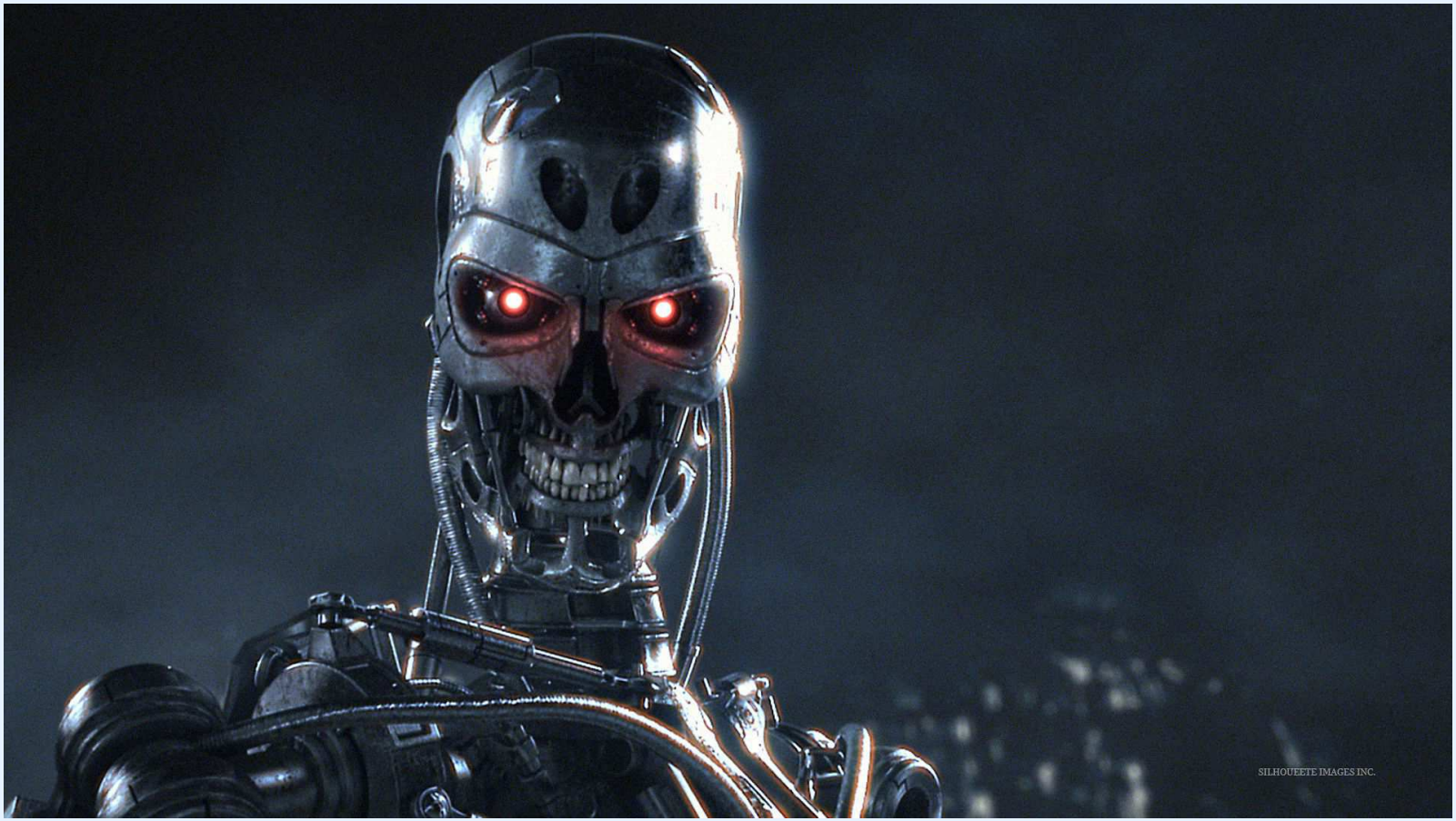
Spark SQL – it's a work in progress.

`https://github.com/HCADatalab/powderkeg/tree/sql`

Spark Streaming – eventually, I don't think there's any work on that yet.

MLlib Machine Learning Library

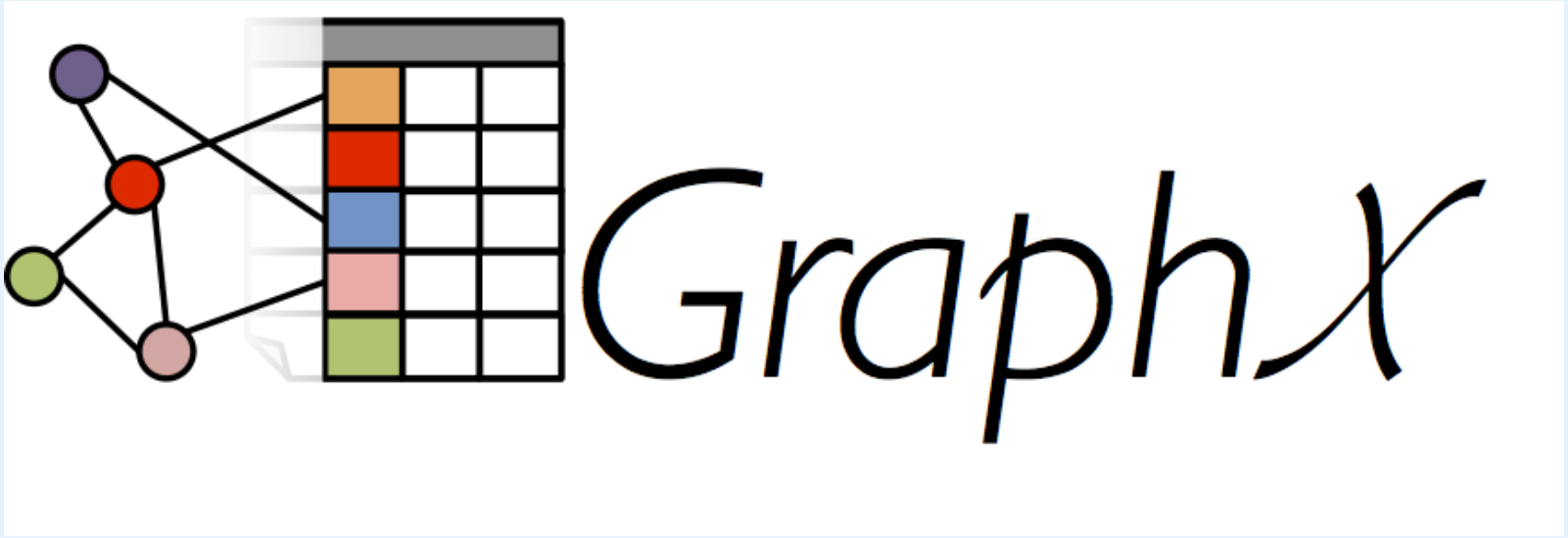
Machine learning is cool. I don't know if anyone is working on that, but I really want it.



MLlib's SVMs, support vector machines, more specifically.

GraphX, a graph processing framework on top of Apache Spark.

I don't know if anyone is working on that, but it looks interesting too.



Questions?