Google

Innovation & Research Infrastructures

A Google perspective

Ed Parsons

Geospatial Technologist, Google Research, London

eparsons@google.com

+ed parsons @edparsons

If I have seen further it is by standing on the shoulders of infrastructures.

1. Hybrid Research at Google

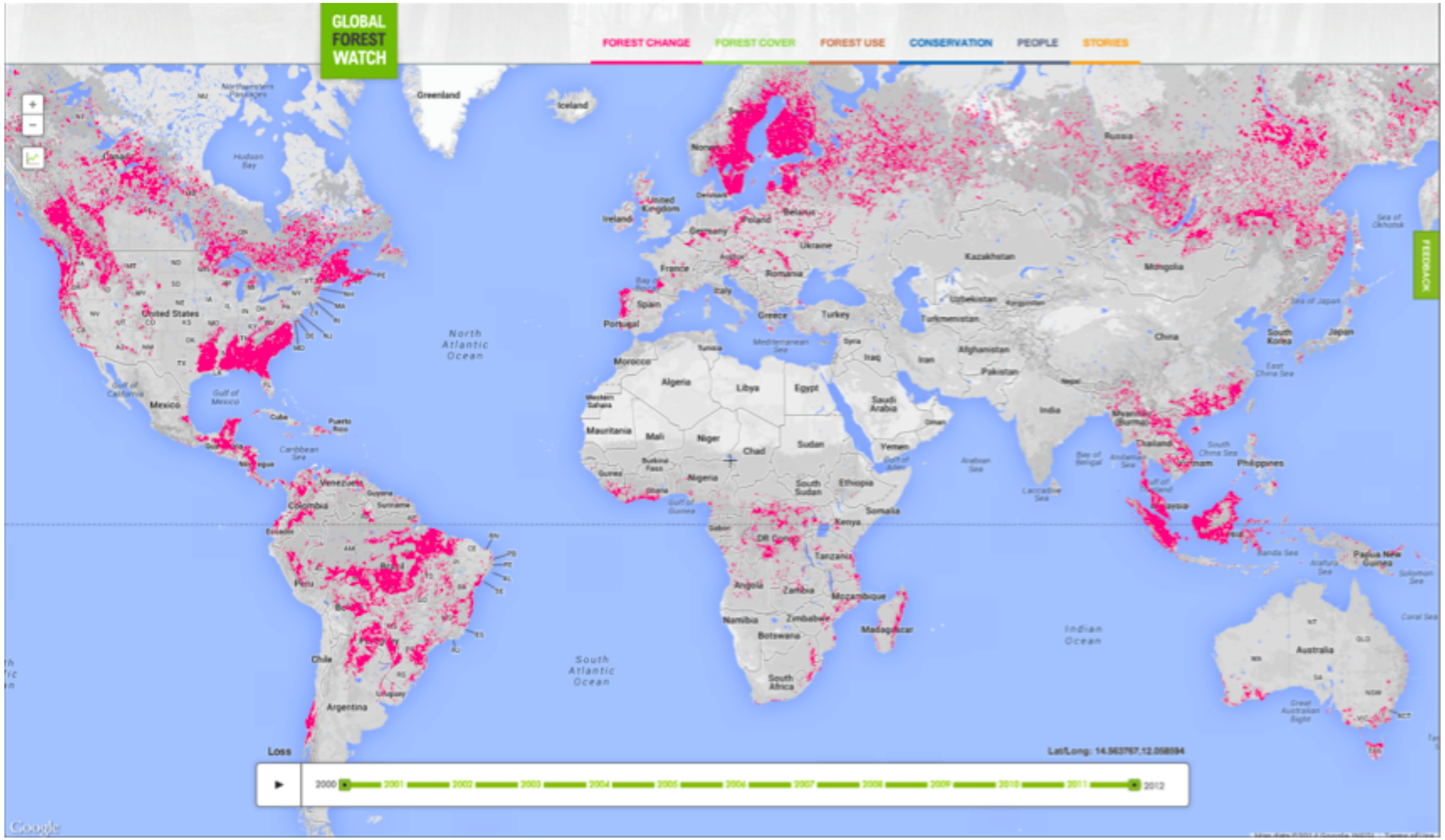
- The goal of research at Google is to bring **significant**, **practical** benefits to our **users**, and to do so rapidly, within a few years at most.
- Approach is iterative and usually involves producing near-production code from day one.
- Single team takes product from Research to Production.
- Requires computing and storage infrastructure at operational scale

2. Research Infrastructures that Scale..

Open,Accessible Platforms

Technology & Cultural

Google

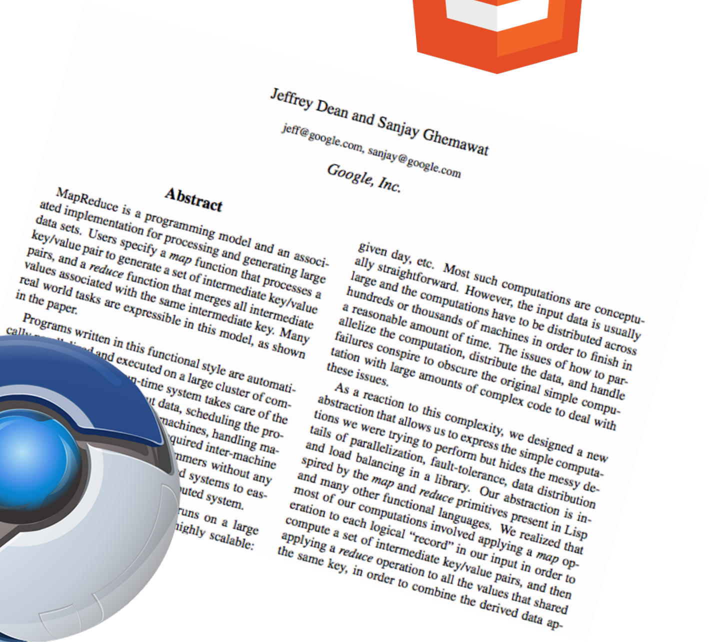


Small teams ⟷ Big Data

Google

3. Measure success..

- Academic, Commercial or ideally both !
- Open Source Projects, Industry standards development



Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

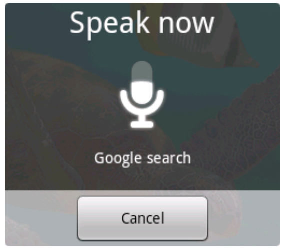
Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automati-
cally [illegible] and executed on a large cluster of com-
[illegible] n-time system takes care of the
[illegible] t data, scheduling the pro-
[illegible] machines, handling ma-
[illegible] quired inter-machine
[illegible] mmers without any
[illegible] d systems to eas-
[illegible] uted system.
[illegible] runs on a large
[illegible] highly scalable:

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a *map* operation to each logical "record" in our input in order to compute a set of intermediate key/value pairs, and then applying a *reduce* operation to all the values that shared the same key, in order to combine the derived data ap-



Ed Parsons

Works at Google
Attended Cranfield University
Lives in Teddington

2,576 have him in circles

www.google.com/+EdParsons

eparsons@google.com

@edparsons