MapReduce

Simplified Data Processing on Large Clusters

Paper by Jeffrey Dan and Sanjay Ghemawat, Google Inc

Outline

- The original MapReduce paper[1] (2004) and the journal version [2]
- Influence of MapReduce paper (2004-2015)
- Beyond MapReduce (2015 - ...)

[1] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." To appear in OSDI (2004): 1.

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

> ***MapReduce is a programming model and an associated implementation for processing and generating large data sets.***
>
> —Literally the first sentence of the paper

Contributions

1. Programming model
2. MapReduce implementation (description)

The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs

The restricted programming model

```
map: (k1,v1) => List[(k2,v2)]
reduce: (k2, List[v2]) => List[v3]
```

Inspired by LISP and other functional programming

Classical example

WordCount

Given lines of text

```
function map( name:String, line:String ) = {
  for( word in line )
    emit ( word, 1 )
}

function reduce( word:String, counts:Iterator[Int] ) = {
  sum = 0
  for( count in counts )
    sum += count
  emit ( word, sum )
}
```

Monoid requirements

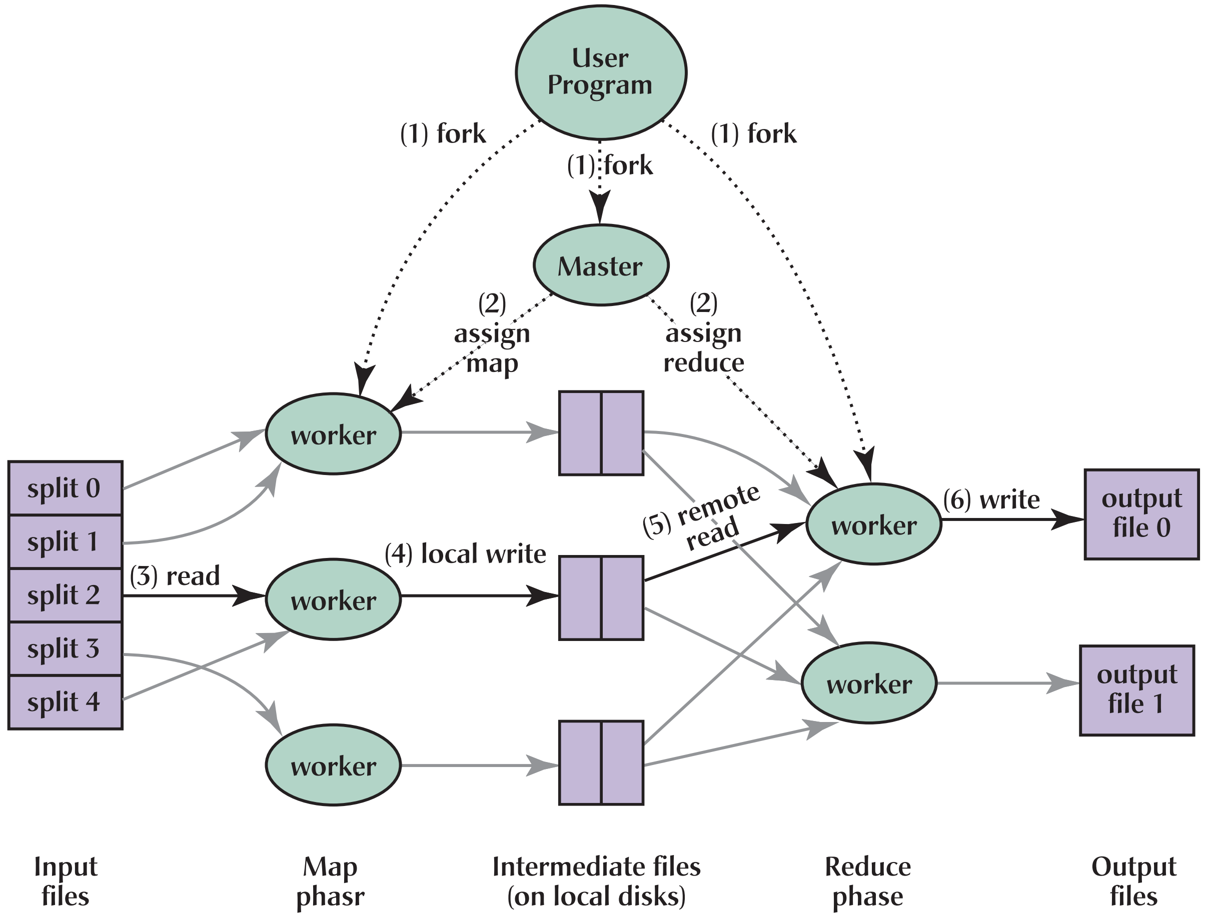
- The reduce function must be associative
  $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- The reduce function must have a neutral element e
  $(x \cdot e) = (e \cdot x) = x, \forall x$

It's still a restricted model

Some operations are not so easy to express in terms of `map` and `reduce`.

The implementation

- Automatic parallelization and distribution
- Fault tolerance
- I/O Scheduling
- Status and monitoring



Taken from [2]

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

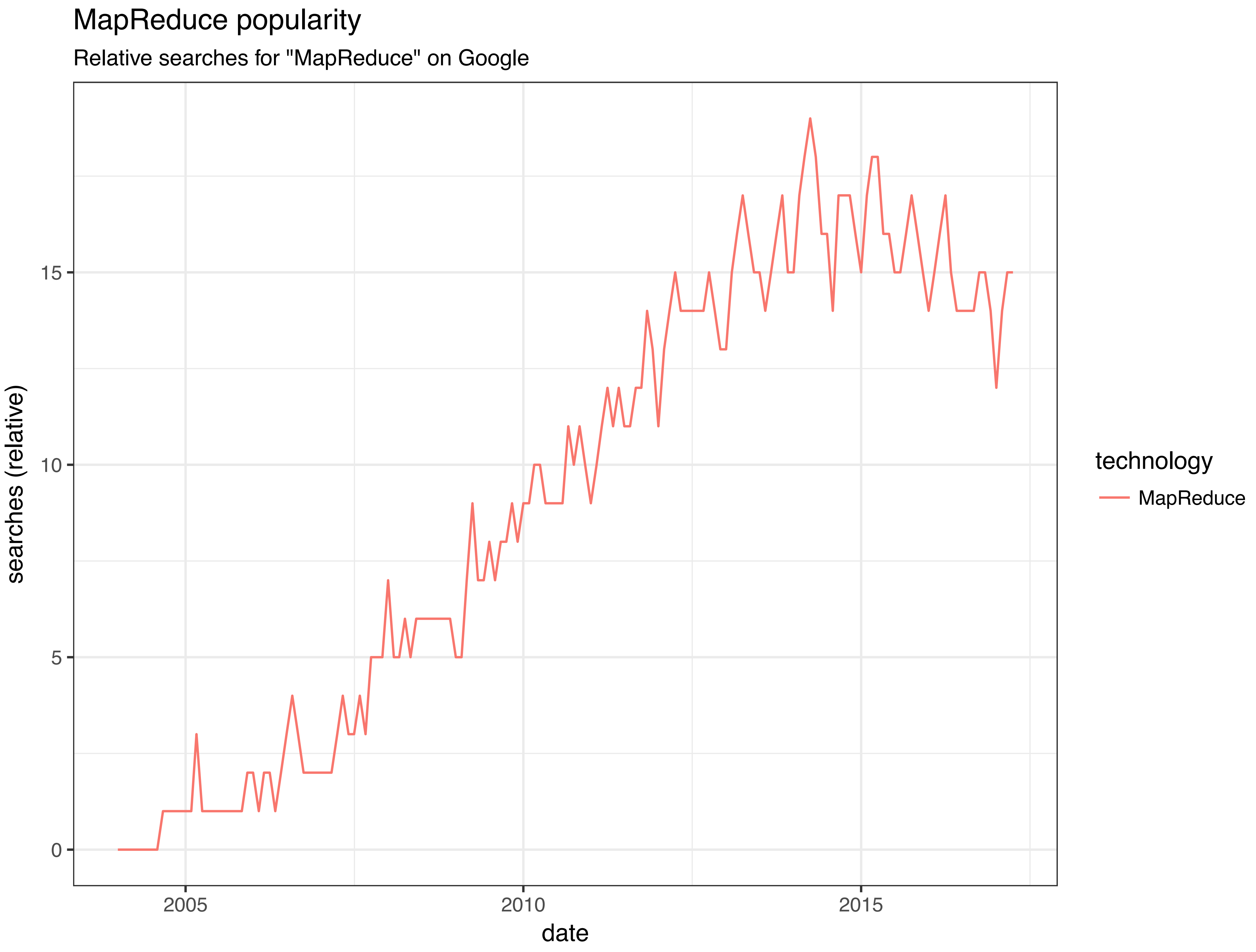
Fault tolerance

- Monitor execution
- Re-execute stale / failed jobs
- Skipping Bad Records

Optimizations

- Partitioning functions
- Combiner functions

Influence of MapReduce paper: Birth of an industry


MapReduce popularity
Relative searches for "MapReduce" on Google
searches (relative)
0
5
10
15
2005
2010
2015
date
technology
MapReduce

Yahoo builds Hadoop

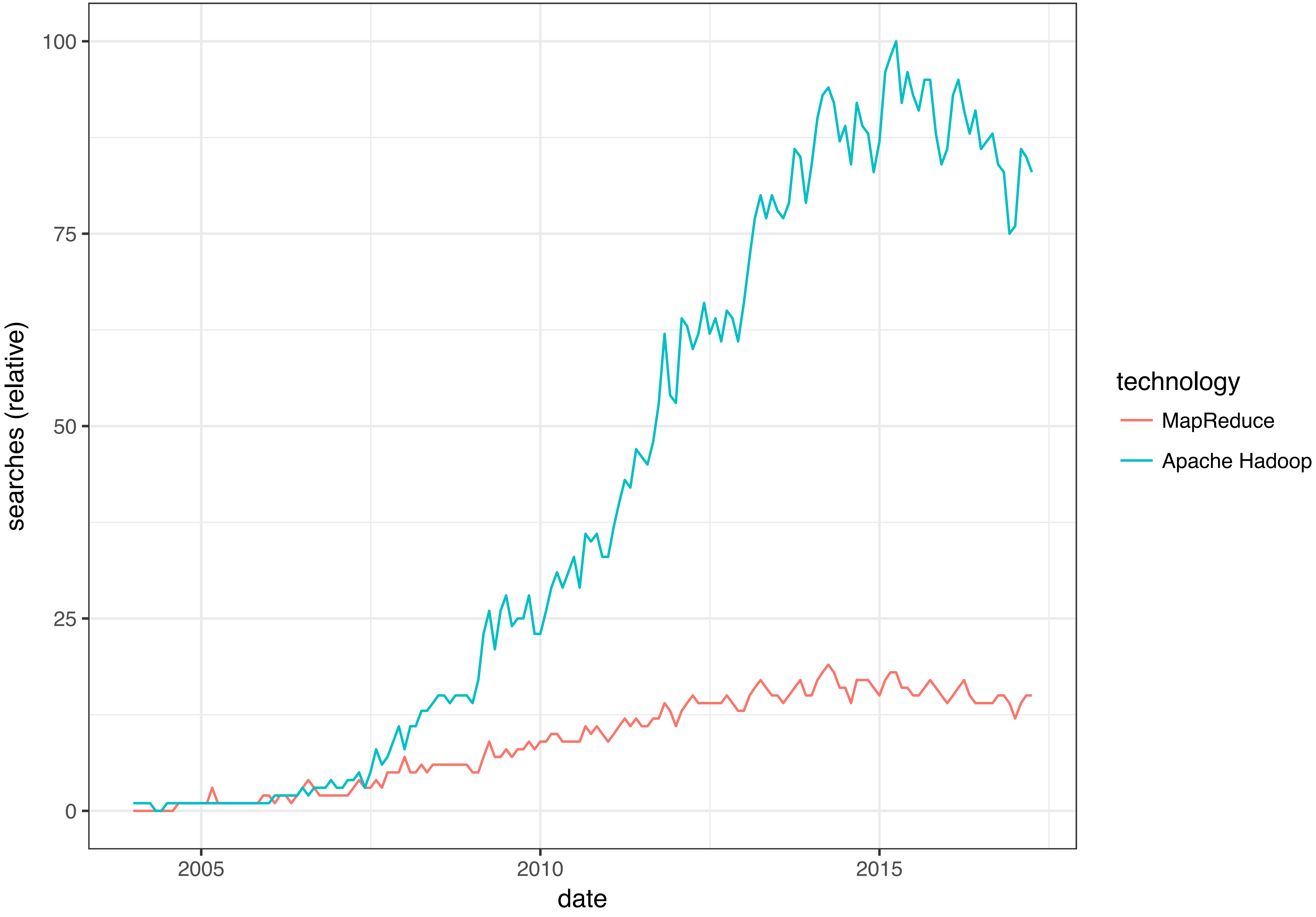
- Implements MapReduce framework based on MapReduce paper[3]
- Implements HDFS based on the GFS paper[4]

[3] Hadoop: Open Source implementation of MapReduce

[4] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.

MapReduce popularity

Relative searches for "MapReduce" on Google



Hadoop ecosystem

PIG, Hive, ZooKeeper and others

All *open source*. Most part of the Apache Software Foundation.

Startups: HortonWorks & Cloudera

Beyond MapReduce

MapReduce, the bad parts

- Everything written to disk
  - Slow
  - RAM becomes cheaper (4GB in original paper)
  - Machine Learning workloads
- Very **restricted** programming model
  - Map - Reduce all the things

Some contenders

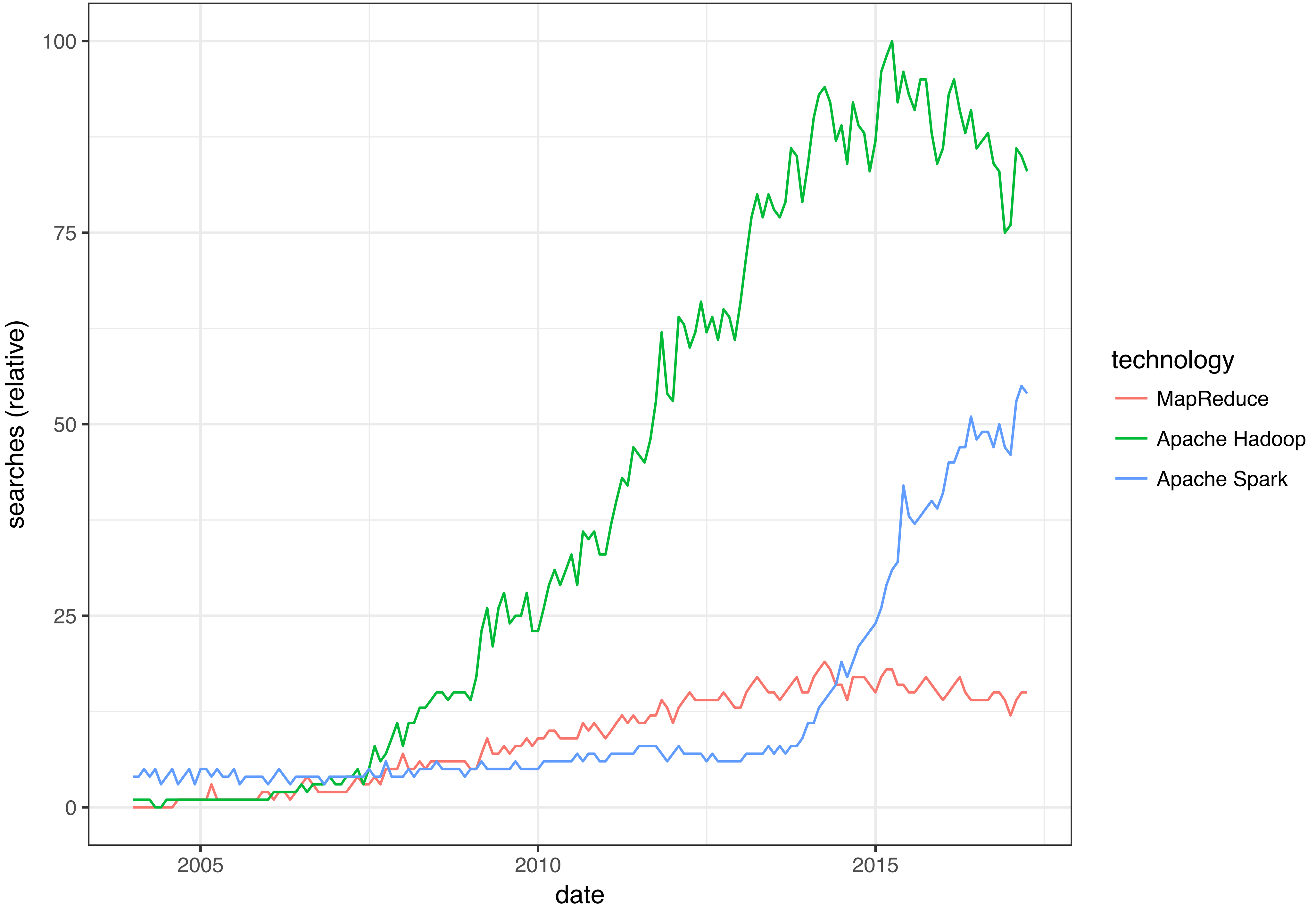
- Cascading
  - Wrapper around MapReduce (Google might have something similar)
- Spark
  - Keep the functional bits, throw out the rest

Spark, the MapReduce killer

- Lazy evaluation
- In memory caching
- More expressive API
- Transition to streaming

MapReduce popularity

Relative searches for "MapReduce" on Google



Classical Example revisited

Wordcount

```
val lines = spark.textFile( "content.txt" )
val counts = lines.flatMap { line =>
  line.split( ' ' ).map( word => ( word, 1 ) )
}.reduceByKey( _ + _ )

println( counts.collect() )
```

Spark, the MapReduce heir

- Still part of Hadoop ecosystem
- Can run on YARN (MapReduce 3.0)
- Very heavily influenced by the functional programming API of Scala

Legacy of MapReduce

Big Data

❤

Functional Programming