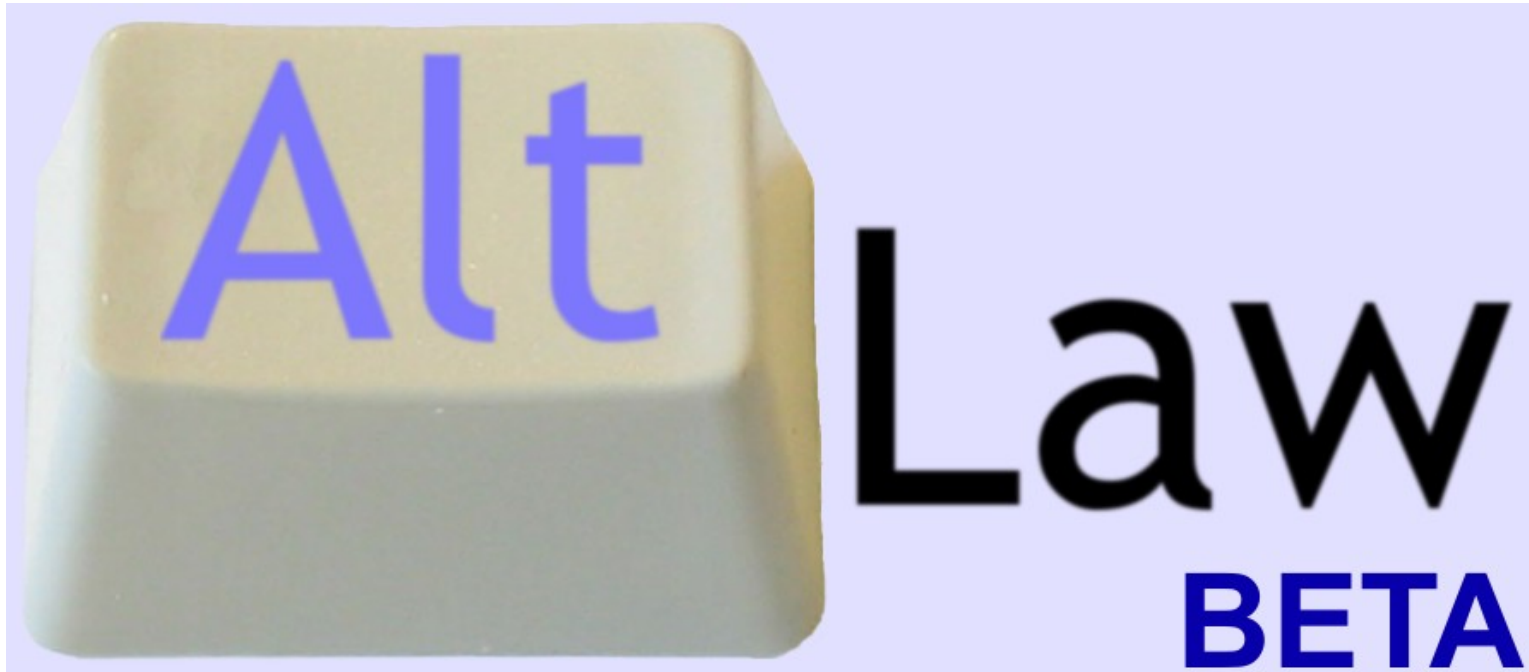


[www.altlaw.org](http://www.altlaw.org)



Stuart Sierra  
Program on Law & Technology  
Columbia Law School  
[www.columbiaLawTech.org](http://www.columbiaLawTech.org)

# Data Sources – Large Corpora

**7 GB**



Paul Ohm, U. of Colorado Law  
[bulk.altlaw.org](http://bulk.altlaw.org)

**200,000 cases**

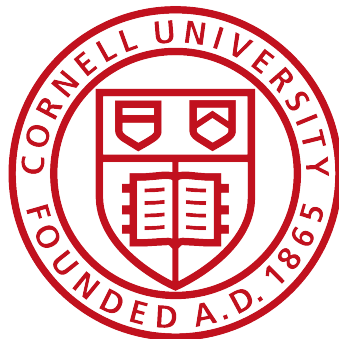
**2 GB**



[public.resource.org](http://public.resource.org)

**700,000 cases**

**0.7 GB**



Cornell Legal Information Institute  
[www.law.cornell.edu](http://www.law.cornell.edu)

**60,000 statutes**

# Data Sources – Court Web Sites

www.supremecourtus.gov  
www.ca1.uscourts.gov  
www.ca2.uscourts.gov  
www.ca3.uscourts.gov  
www.ca4.uscourts.gov  
www.ca5.uscourts.gov  
www.ca6.uscourts.gov

...

**14** appeals courts total

**94** district courts

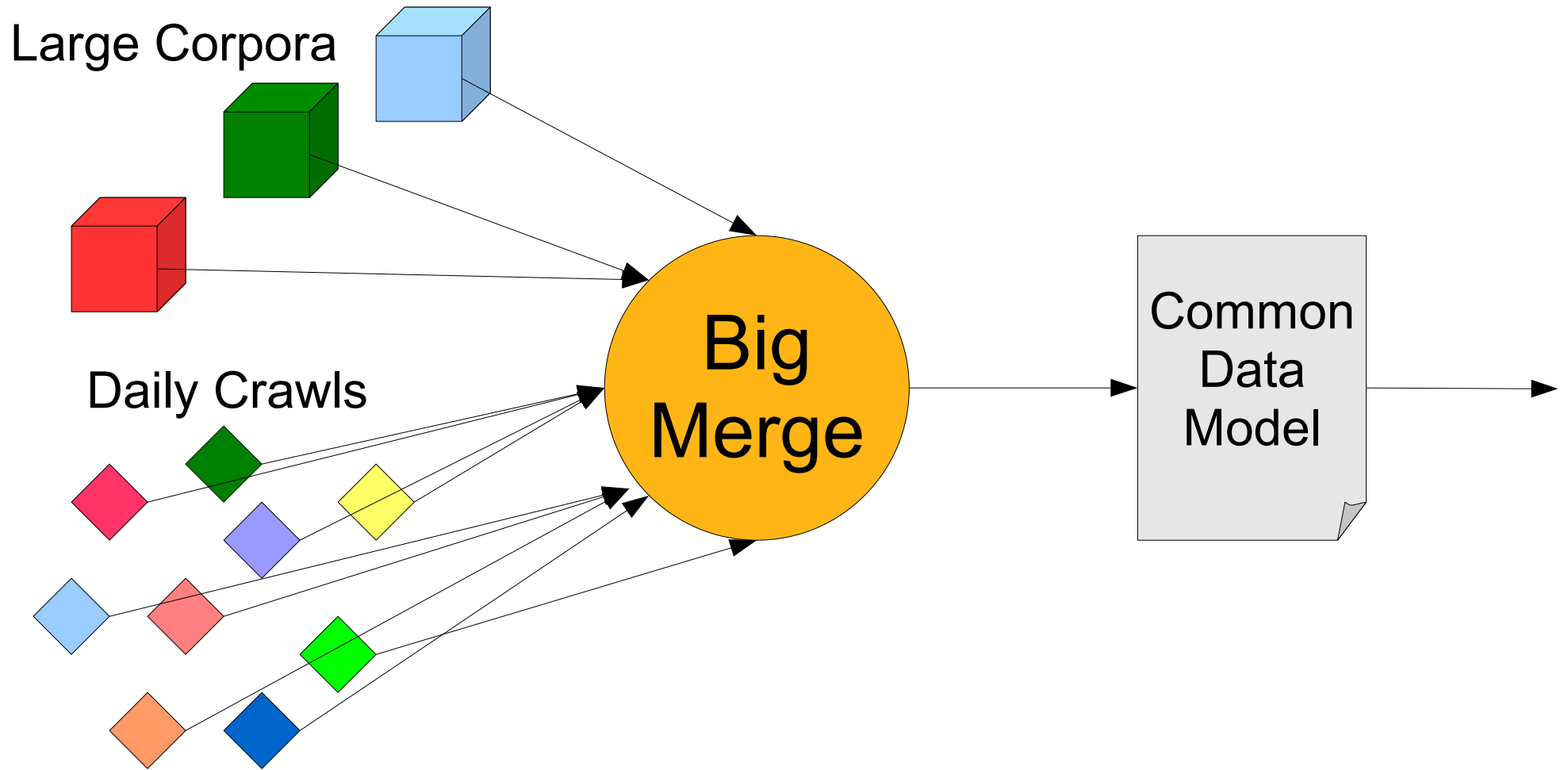
?? state courts

?? local/other courts

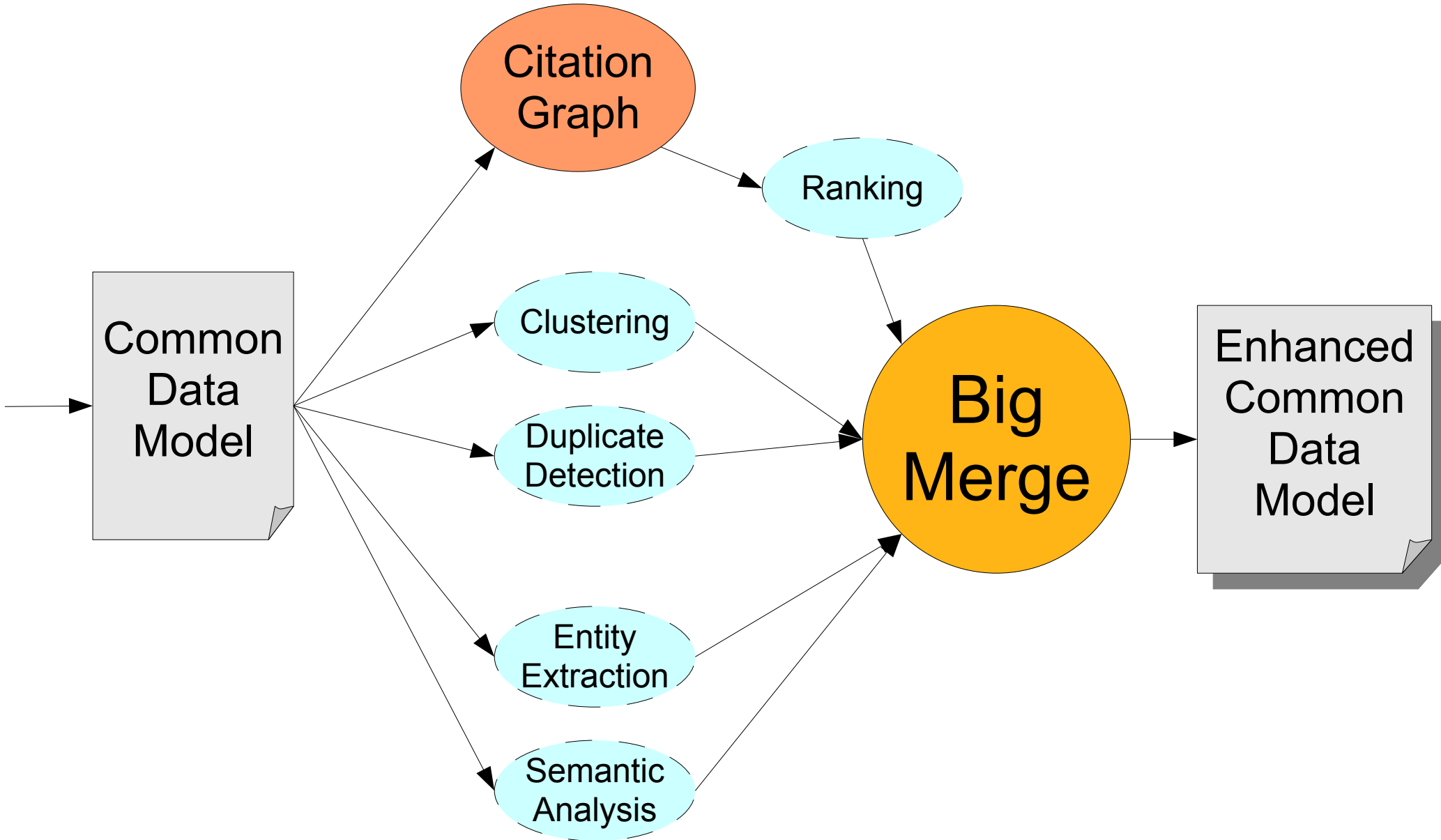
- 20-40 new cases daily
- PDF, WordPerfect, HTML, plain text

The screenshot shows a Mozilla Firefox browser window displaying the website for the United States Court of Appeals for the Second Circuit. The browser's address bar shows the URL <http://www.ca2.uscourts.gov/opir>. The website header includes the court's name, Daniel Patrick Moynihan U.S. Courthouse, 500 Pearl Street, New York, NY 10007, and phone number (212) 657-8500. Below the header, the names of the Chief Judge (Dennis Jacobs) and Circuit Executive (Karen Greve Milton) are listed. The main content area features a navigation menu on the left with buttons for Home, Annual Reports, Attorney Admissions, Attorney Discipline, C.A.M.P., C.J.A., Court Calendars, Decisions, Dockets, and Forms. The main content area is titled "OPINIONS" and includes filters for "Today's", "Last 7 Days'", "Last 14 Days'", and "Last 30 Days'". Below this is a "SUMMARY ORDERS" section with similar filters. The "SEARCH ALL" section includes instructions: "Docket Number should be entered as #-####", "Date should be entered as mm/dd/yyyy", and "You must select a database to search". The search form has a "Search in:" dropdown menu set to "Opinions Only", a "Sort Results By:" dropdown menu set to "Date posted", and a "Results Per Page:" dropdown menu set to "10". At the bottom, there is a prompt: "Enter Docket #, Date, Party Name".

# Back-end (1)



# Back-end (2)



# MapReduce – it works for Google

- Disk is the bottleneck.
- *Seek time* is the bottleneck.

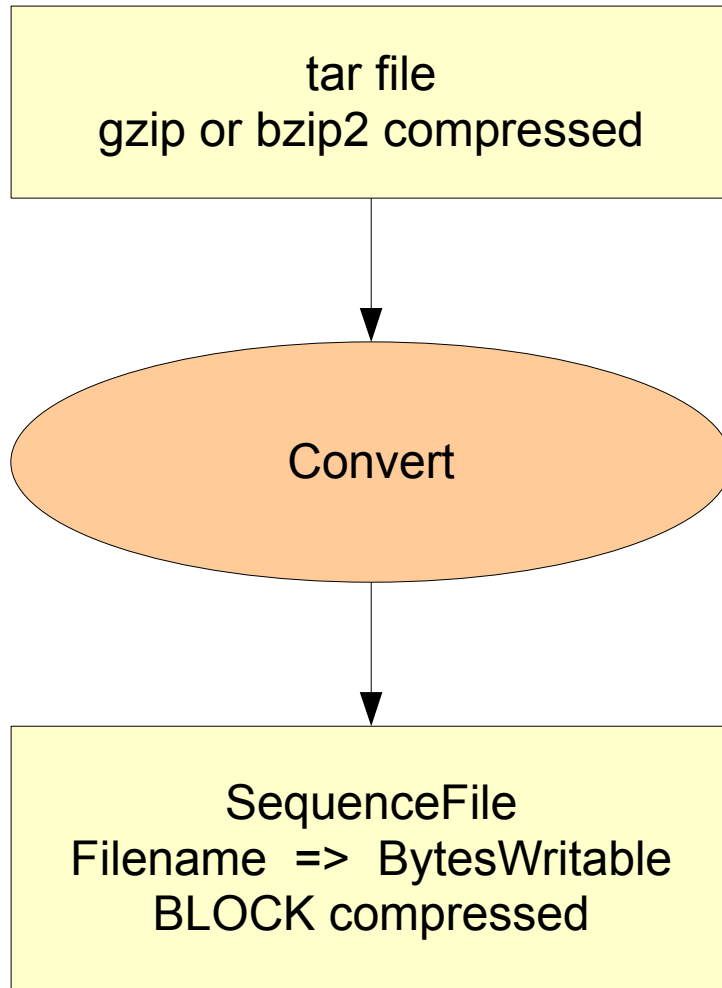
MapReduce:

- Minimize disk seeks.
- Run at full transfer rate.

**“Disk is the new tape.”**

# The Million Little Files Problem

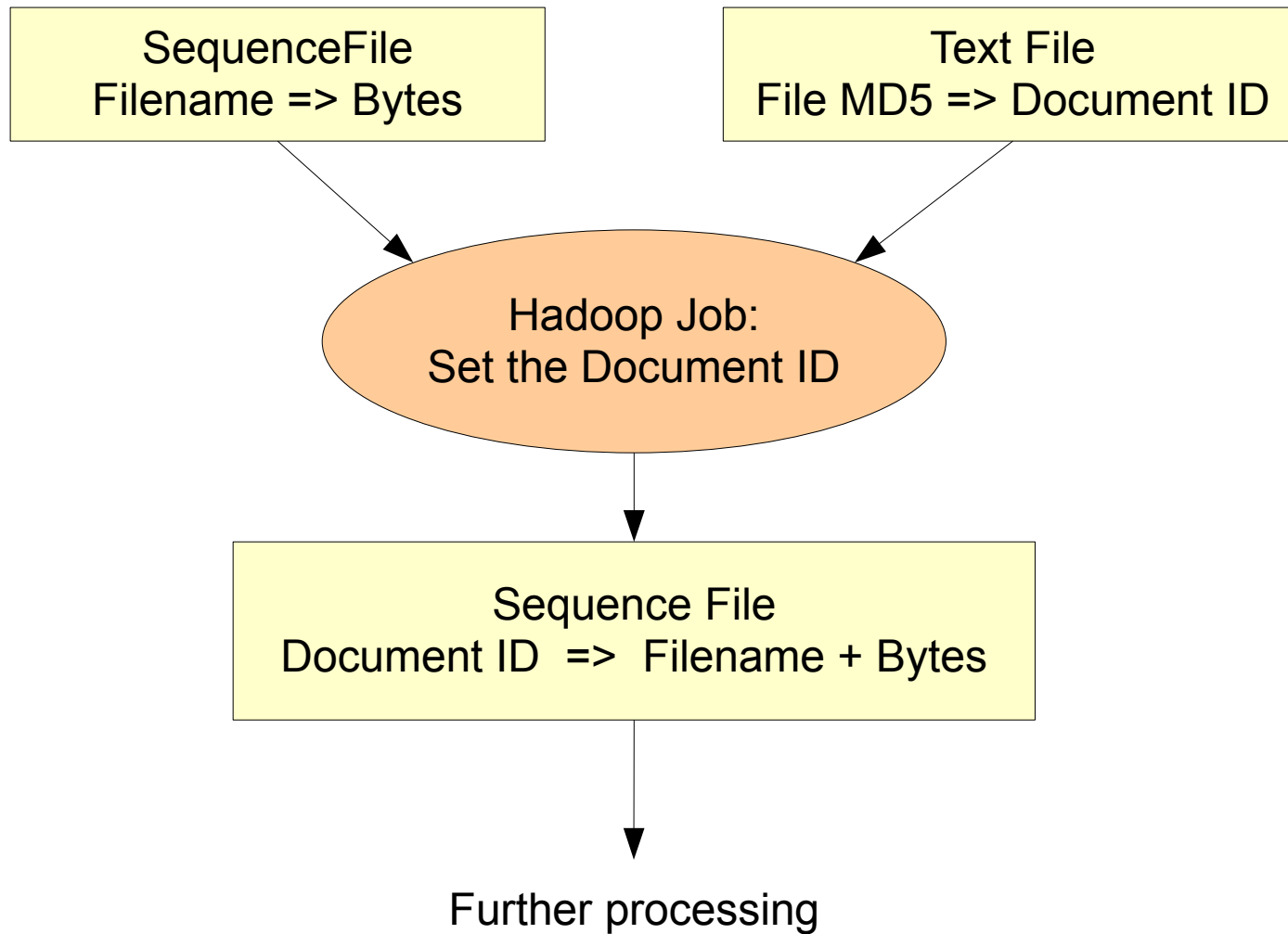
<http://stuartsierra.com/2008/04/24/a-million-little-files>



- Cannot run in parallel
- Slow: 20 MB / minute
- Only has to be done once

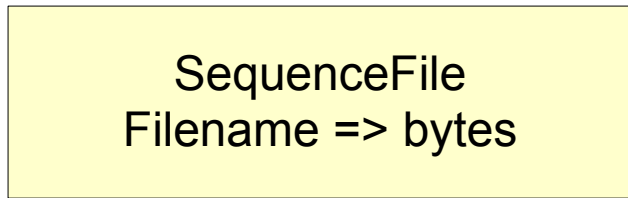
Newer solution in 0.19:  
Hadoop Archives (HAR files)

# The Document ID Problem

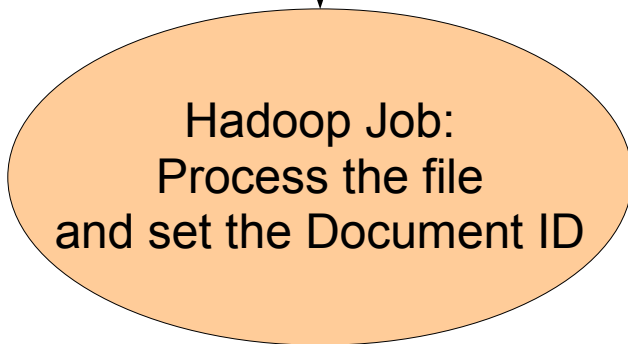
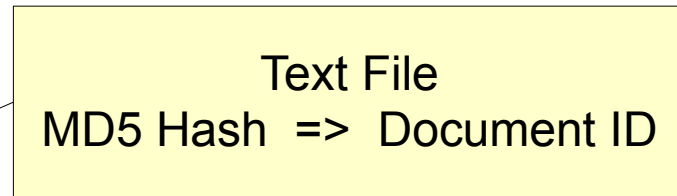


# If it Fits in RAM, Keep it in RAM

2 GB



26 MB

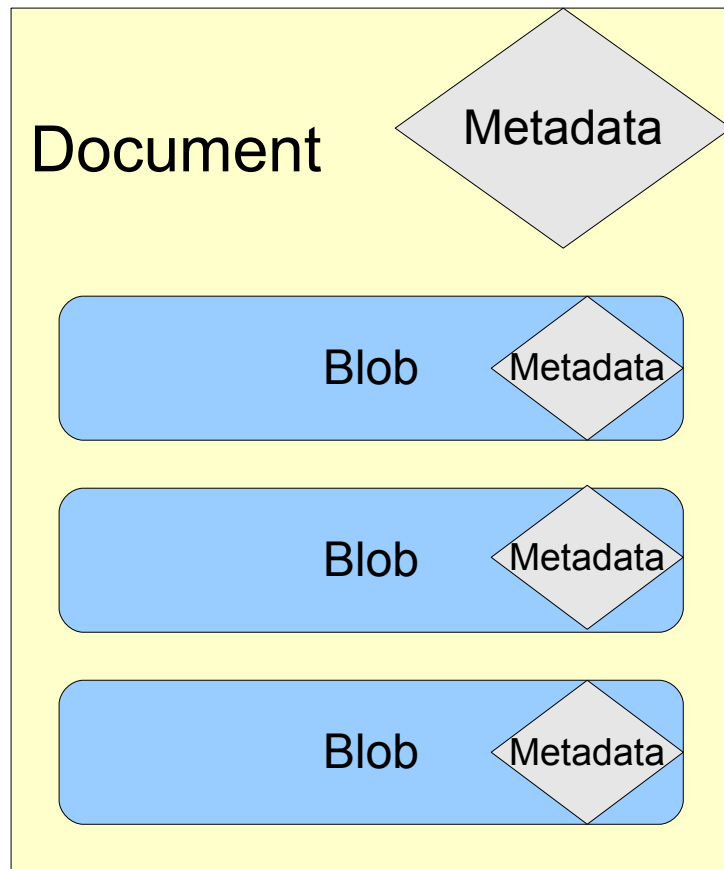


Copy file to DistributedCache.  
Load once in Mapper.configure().  
Store in a HashMap.

Further processing

# The State Management Problem

## The Document-Blob Model



or, the “Store Everything” model.

or, Stuart's worst idea ever.

# The State Management Problem

- “Out of the Tar Pit”

<http://lambda-the-ultimate.org/node/1446>

## Essential State

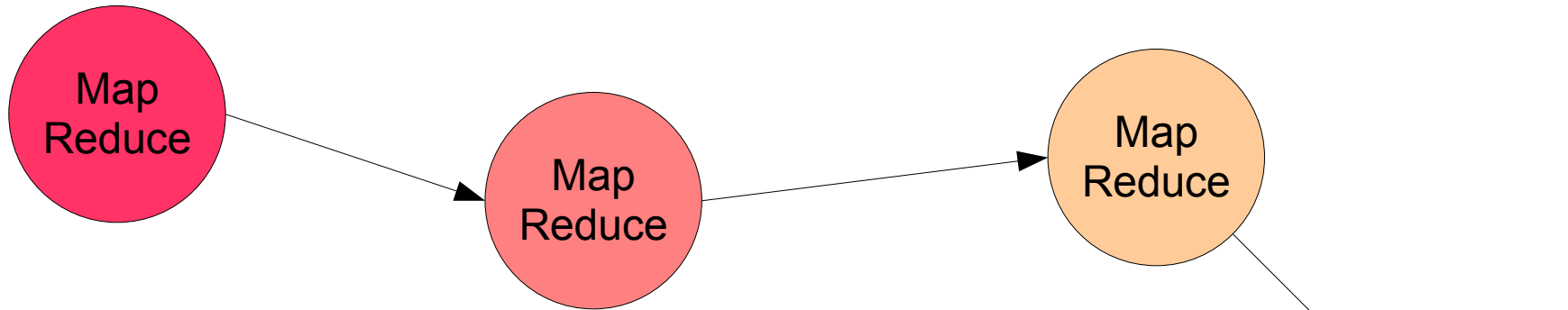
You have to store this.

## Derived State

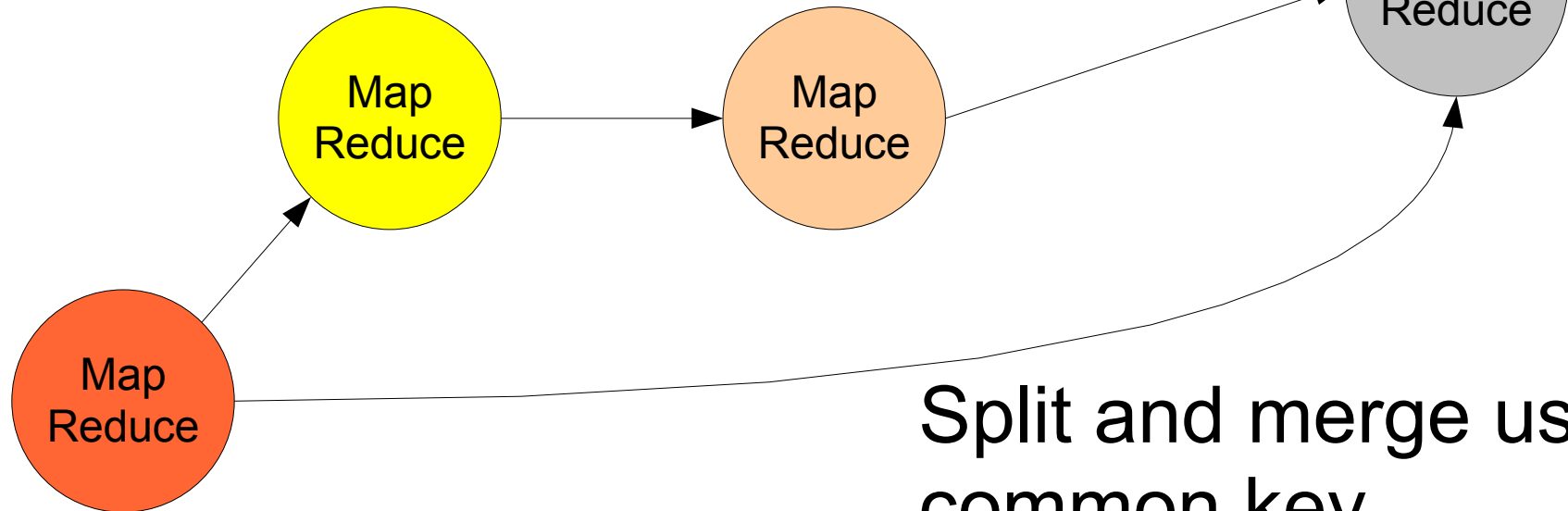
Never store this.

Cache as needed  
for performance.

# The State Management Problem

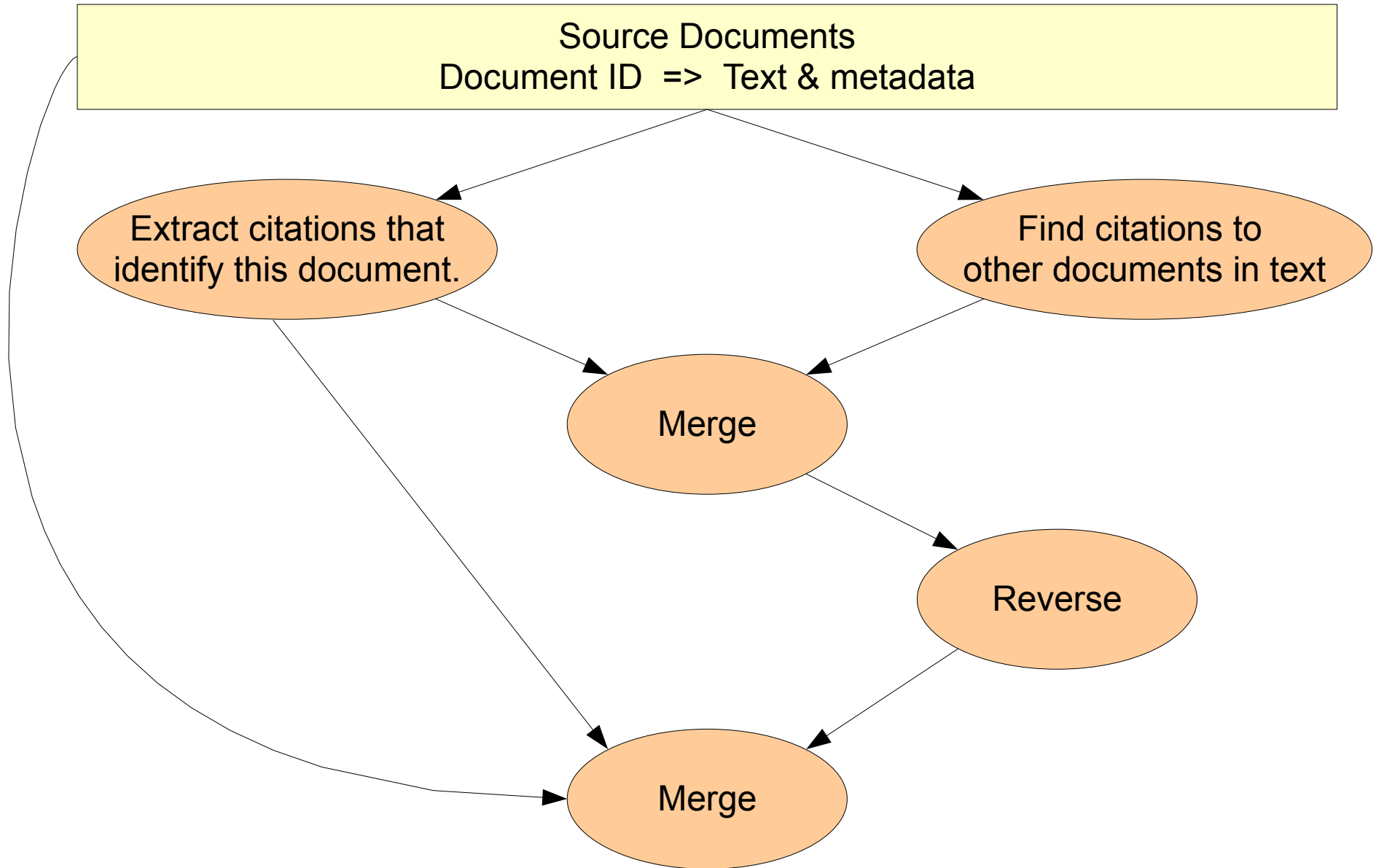


State is implicit in the order of jobs.

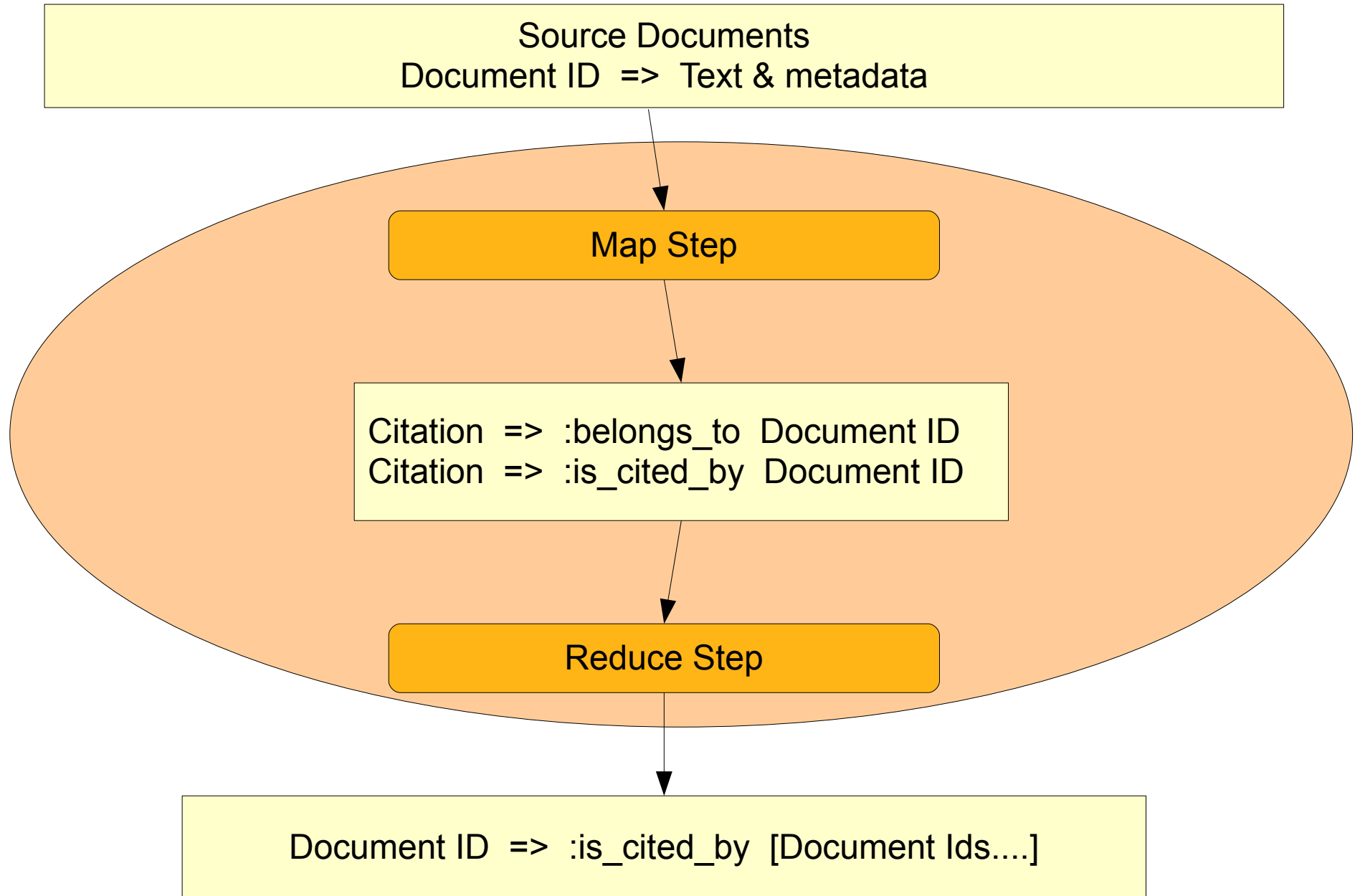


Split and merge using common key.

# The Citation Linking Problem



# The Citation Linking Problem



[www.clojure.org](http://www.clojure.org)

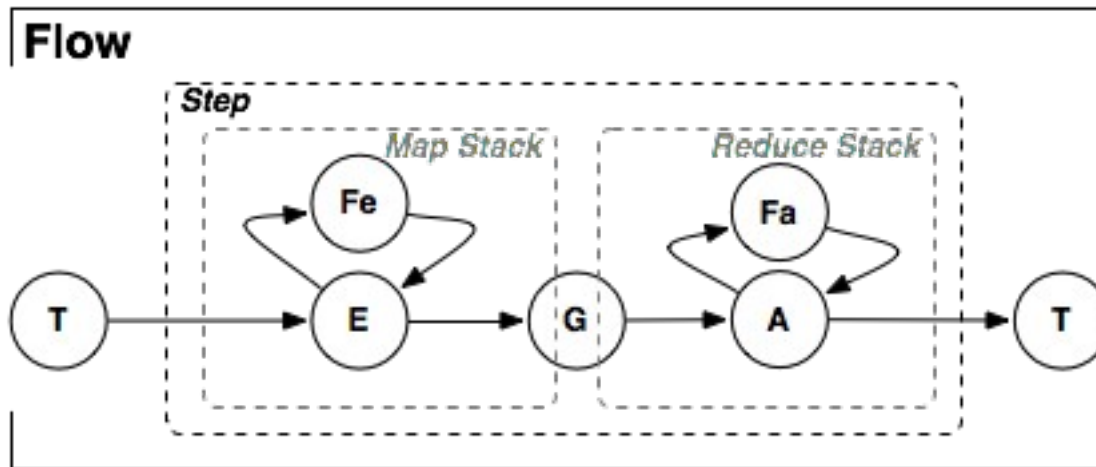
```
(ns org.altlaw.jobs.link.linkincites-mapred
  (:gen-class
   :extends org.apache.hadoop.mapred.MapReduceBase
   :implements [org.apache.hadoop.mapred.Mapper
                org.apache.hadoop.mapred.Reducer])
  (:use org.altlaw.util org.altlaw.util.hadoop)
  (:refer clojure.set))

(import-hadoop)

(defn -map [this wkey wvalue output reporter]
  (.collect output wkey
    (if (instance? Text wvalue)
        wvalue
        (Text. (str wvalue)))))

(defn -reduce [this wdocid ivalues output reporter]
  (let [values (doall (map (comp read-string str) (iterator-seq ivalues)))
        cited-docids (filter integer? values)
        citing-doc (select-keys
                     (first (filter (complement integer?) values))
                     [:docid :doctype :citations :court :date :name])]
    (doseq [docid cited-docids]
      (.collect output (IntWritable. docid)
        (Text. (pr-str {:incites #{citing-doc}}))))))
```

# Cascading



# What I Learned

- Clojure + Hadoop = awesome.
- If it fits in RAM, keep it in RAM.
- Don't store what you can recompute.
- Plan to run every job repeatedly.
- Design jobs to be chained.
- A job can have heterogeneous output.
- Test on small samples.