L24: MapReduce and DFS

Jeff M. Phillips

April 16, 2018

Poster

Don't put too much, or small font!

- Succinct title (and names)
- What is the problem and data you worked on?
- What were the key ideas in your approach?
- What techniques from the class did you use?
- What did you learn?

Big Data ⟷ Data Science

Classic Scientific Paradigm

Data Science

⓪ Exists large data set ⟹ ① Search for hypothesis/pattern supported by data.

Distributed File System (HDFS)

everything: <key, value> pair.

key	value
log id	log
web address	html, out-going-links
doc id	list of words
word	how many times (and where) it appears in doc.

Really big data → store in blocks (64 MB)

- 2-3 copies of each blocks.
- store each copy on different computers. → at random.

Why little locality

- Resiliency: if computer goes down, another that can take its place.
- Redundancy: split read some across cluster.
- Heterogeneous: different type of data.

infrequent updates
mainly appends.

Map Reduce

Queries over DFS

usually, output is large.

1. Map: Transformation $\langle k, v \rangle \to \langle k', v' \rangle$
   when its useful to group/sort on k'

1.5. Combine (Optional): Preprocessing $\langle k', v_1 \rangle$ $\langle k', v_2 \rangle$ on same computer before shuffle.

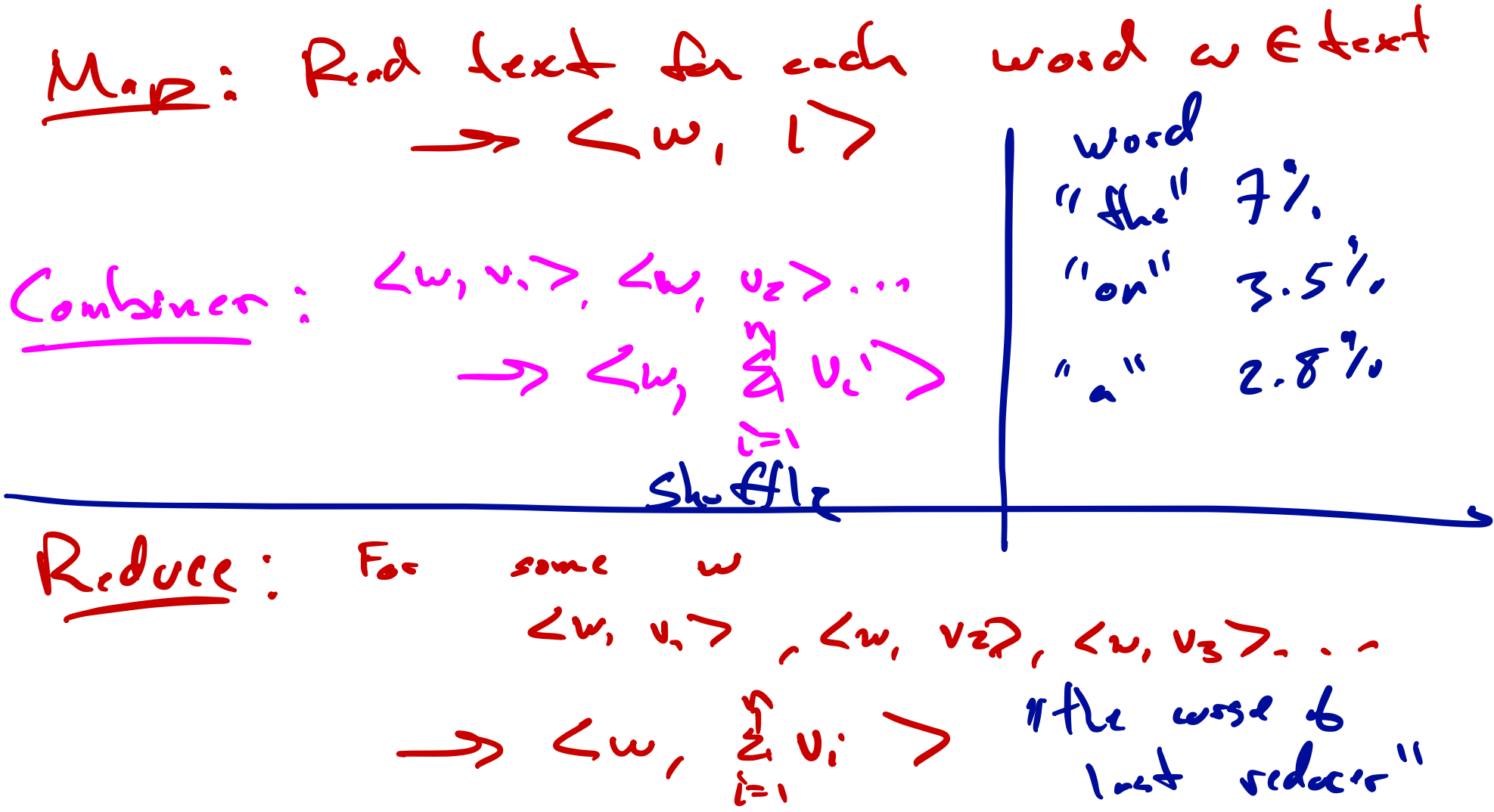
2. Shuffle: Put all $\langle k', v_1 \rangle$ and $\langle k', v_2 \rangle$ with same k' on same computer.

3. Reduce: Take $\langle k', v_1 \rangle$ $\langle k', v_2 \rangle$ ...
   Map to set $\langle k'', V = f(v_1, v_2, v_3, \ldots v_n) \rangle$

Word Count

Need at least 20, 50 computers.

Consider as input all of English Wikipedia stored in DFS. Goal is to count how many times each word is used.



Rounds

Rounds are slow.

Overhead. + write everything to disk each round.

↳ Minimize # rounds

$\log_2 N$ too many.

≈ 30

⟹ Spark (Berkeley)

Inverted Index

Consider as input all of English Wikipedia stored in DFS. Goal is to build an index, so each word has a list of pages it is in.

Map: For all words w on page p.

$\hookrightarrow \langle w, p \rangle$

Reduce: $\langle w, p_1 \rangle, \langle w, p_2 \rangle, \ldots$

$\hookrightarrow \langle w, \bigcup_{i=1}^{n} p_i \rangle$

↑ maybe sort by pageRank.

Phrases

Consider as input all of English Wikipedia stored in DFS. Goal is to build an index, on 3-grams (sequence of 3 words) that appears on exactly one page, with link to page.

w = 3-gram

Map → $\langle w, p \rangle$

Combine : $\langle w, p \rangle, \langle w, p \rangle \rightarrow \langle w, p \rangle$

Reduce : if $\langle w, p_1 \rangle, \langle w, p_2 \rangle \ldots$

has > 1 p_i → ϕ

o.w. $\langle w, p \rangle$

Page Rank

Probability transition matrix M

column M_i = $\begin{bmatrix} 0 \\ 1/2 \\ \vdots \\ 0 \\ 1/m \end{bmatrix}$ ← index of page j

i links to out of k pages

$\beta = 0.85$

$$P = \beta M + (1-\beta)B \qquad B = \begin{bmatrix} 1/m & 1/m & \cdots \\ 1/m & 1/m & \cdots \end{bmatrix} \downarrow m \rightarrow m$$

$$q_{i+1} = P q_i = \left(\beta M + (1-\beta)B\right) q_i$$

Block (set of web pages)
(= set of columns M)

Map: Block $M[a,b]$, query q_i

$q_i[a,b]$ $\langle k, q_{i+1} = M[a,b] \cdot q_i[a,b] \rangle$

shuffle

Reduce $q_{i+1} = \beta \cdot \sum_{j=1}^{n} (q_{i+1})_j + (1-\beta) \begin{bmatrix} \frac{1}{n} \\ \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{bmatrix}$

Example PageRank

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

out webpage

Example PageRank

what if
q_i too big
for 1 comp.

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Stripes:

$$M_1 = \begin{bmatrix} 0 \\ 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \quad M_2 = \begin{bmatrix} 1/2 \\ 0 \\ 0 \\ 1/2 \end{bmatrix} \quad M_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad M_4 = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 0 \end{bmatrix}$$

These are stored as $\big(1 : (1/3, 2), (1/3, 3), (1/3, 4)\big)$,
$\big(2 : (1/2, 1)(1/2, 4)\big)$, $\big(3 : (1, 3)\big)$, and $\big(4 : (1/2, 1), (1/2, 3)\big)$.

Example PageRank

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Blocks:

$$M_{1,1} = \begin{bmatrix} 0 & 1/2 \\ 1/3 & 0 \end{bmatrix} \quad M_{1,2} = \begin{bmatrix} 0 & 0 \\ 1 & 1/2 \end{bmatrix} \quad M_{2,1} = \begin{bmatrix} 1/3 & 0 \\ 1/3 & 1/2 \end{bmatrix} \quad M_{2,2} = \begin{bmatrix} 0 & 1/2 \\ 0 & 0 \end{bmatrix}$$

These are stored as $\big(1 : (1/2, 2)\big), \big(2 : (1/3, 1)\big)$, as $\big(2 : (1, 3), (1/2, 4)\big)$, as $\big(3 : (1/3, 1)\big)$, $\big(4 : (1/3, 1), (1/2, 2)\big)$, and as $\big(3 : (1/2, 4)\big)$.