Distributed Data Management
Summer Semester 2013
TU Kaiserslautern

Dr.-Ing. Sebastian Michel

smichel@mmci.uni-saarland.de
Oral Exams Dates

• **Note: Last week of teaching** at University, SS’13
  – July 15 - July 20, 2013

• How about: all (around 5?) slots in July (after end of teaching, or in last week) or early August.

• Your preferences?
Lecture 4

MAP REDUCE: APPLICATIONS (CONT’D)
Recap: Map and Reduce

- Map \((k1,v1) \rightarrow \text{list}(k2,v2)\)
- Reduce\((k2, \text{list}(v2)) \rightarrow \text{list}(k3, v3)\)

- For instance:
  - \(k1=\text{document identifier}\)
  - \(v1=\text{document content}\)
  - \(k2=\text{term}\)
  - \(v2=\text{count}\)
  - \(k3=\text{term}\)
  - \(v3=\text{final count}\)

keys allow grouping data to machines/ tasks
(Equi) Join of 3 Relations

\[ R(A,B) \text{ Join } S(B,C) \text{ Join } T(C,D) \]

Can be implemented as

- Two 2-way joins, e.g., \( R(A,B) \text{ Join } S(B,C) \) and then the result joined with \( T(C,D) \)
- Or directly, how?
Join of 3 Relations: Considerations

\[ R(A,B) \Join S(B,C) \Join T(C,D) \]

• Send tuples of \( S \) by key \((b,c)\), but tuples in \( R \) and \( T \) for many combinations \((*,b)\) and \((c,*)\)

• Note: Theta joins (with arbitrary join predicate) are much more complicated.

Foto N. Afrati, Jeffrey D. Ullman: Optimizing joins in a map-reduce environment. EDBT 2010: 99-110
Alper Okcan, Mirek Riedewald: Processing theta-joins using MapReduce. SIGMOD Conference 2011: 949-960
n-Grams

• Statistics about **variable-length word sequences**
  (e.g., lord of the rings, at the end of, …) have **many applications** in fields including
  – Information Retrieval
  – Natural Language Processing
  – Digital Humanities

• E.g., [http://books.google.com/ngrams/](http://books.google.com/ngrams/)

• A n-gram dataset is also available from there

Distributed Data Management, SoSe 2013, S. Michel
n-gram slides based on a talk by Klaus Berberich
Example: Google Books Ngrams
n-grams Example

- **Document**: a x b b a y
- **Possible n-grams**:
  - (a), (x), (b), (y)
  - (ax), (xb), (bb), ...
  - (axb), (xbb), ...
  - (axbb), (xbba), (bbay)
  - (axbba), (xbbay)
  - (axbbay)
Task: Computing n-grams in MR

• How can we efficiently compute n-grams, that
  – occur at least $\tau$ times
  – and consist of at most $\sigma$ words

using MapReduce?
Naïve Solution: Simple Counting

\texttt{map}(\texttt{did, content}): \\
\hspace{1cm} \textbf{for} \ k \ \text{in} \ \langle 1 \ \ldots \ \sigma \rangle: \\
\hspace{2cm} \textbf{for all} \ \text{\textit{k-grams in content}}: \\
\hspace{3cm} \texttt{emit}(k\text{-gram, did})

\texttt{reduce}(\texttt{n-gram, list<did>}): \\
\hspace{1cm} \textbf{if} \ \text{\text{\texttt{length}}(\texttt{list<did>})} \geq \ \tau: \\
\hspace{2cm} \texttt{emit}(n\text{-gram, length(list<did>))}
A Priori Based

• **A priori Principle**: $k$-gram can occur more than $\tau$ times only if its constituent $(k-1)$-grams occur at least $\tau$ times

  \[(a,b,c) \text{ qualified only if } (b,c), (a,b) \text{ and } (a), (b), (c)\]

How to implement?

*) Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference 1993: 207-216
A Priori Based (Cont’d)

• Iterative Implementation:
  – First 1-grams that occur \( \tau \) times
  – Then 2-grams that occur \( \tau \) times
  – ...

• Needs multiple MapReduce rounds (of full data scans)

• Already determined \( k \)-grams are kept
Suffix Based

• Emit only suffixes in map phase
• Each of them represents multiple $n$-grams corresponding to its prefixes
  – For instance, axbbay represents
    • a, ax, axb, axbb, axbba, and axbbay

map(did, content):
  for all suffixes in content:
    emit(suffix, did)
Suffix Based: Partitioning

• Partition the suffixes by first word
  – to ensure all n-grams end up property for counting, that is:
    • all occurrences of ax have to end up at same reducer
    • suffix property: ax is only generated from suffixes that start with ax..

\[
\text{partition}(\text{suffix}, \text{did}):\]

\[
\text{return } \text{suffix}[0] \ % \ m
\]
Suffix Based: Sorting

• Reducer has to generate n-grams based on suffixes
  – read prefixes
  – count for each observed prefix its frequency
  – optimization: sort suffixes in reverse lexicographic order
  – then: simple counting using stack

```
compare(suffix0, suffix1):
    return -strcmp(suffix0, suffix1)
```
Discussion

• Assess aforementioned algorithms with respect to properties like:
  – multiple MapReduce jobs vs. single job
  – amount of network traffic
  – ease of implementation
GRAPH PROCESSING IN MAPREDUCE
Graph Processing in MapReduce

• General: Graph Representation
  – usually: Adjacency list
  • \( v_1 \rightarrow v_2, v_4, v_5 \)
  • \( v_2 \rightarrow v_4 \)
  • \( v_3 \rightarrow v_5 \)
  • ...
Refresher: Breadth First Search (BFS)

Q = FIFO queue
enqueue start node

while not found:
    n := Q.dequeue
    if n== target then break
    foreach c in n.childlist
        Q.enqueue(c)

• Example visiting order:

Distributed Data Management, SoSe 2013, S. Michel
Graph Processing in MapReduce

• No global state in MapReduce
• Need to pass on results AND graph structure

```java
map(id, node) {
    emit(id, node)
    partial_result = local_compute()
    for each neighbor in node.adjacencyList {
        emit(neighbor.id, partial_result)
    }
}
```
Graph Processing in MapReduce (2)

```java
reduce(id, list) {
  foreach msg in list{
    if instanceof(msg) == Node
      node = msg
    else
      result = aggregate(result, msg)
  end
  node.value = result
  emit(id, node)
}
```
BFS in MapReduce

• How to implement Breadth First Search in MapReduce?

• Hint: Need to pass on structure (as seen) before. Augment nodes with additional information: visited, distance.
Application: Computing PageRank

- Link analysis model proposed by Brin & Page
- Compute authority scores
- In terms of:
  - incoming links (weights) from other pages
- “Random surfer model”

PageRank: Formal Definition

• PageRank of a page q:

\[ PR(q) = \varepsilon \times \sum_{p : p \rightarrow q} \frac{PR(p)}{\text{out}(p)} + (1 - \varepsilon) \times \frac{1}{N} \]

- \( N \) → Total number of pages;
- \( PR(p) \) → PageRank of page p;
- \( \text{out}(p) \) → Outdegree of p
- \( \varepsilon \) → Random jump probability

• Iterative computation until convergence

• *Dangling nodes*: “Sinks”. Solution: Add random jump (uniform) to any other nodes.
Formal Model of Web Graph

• Matrix representation of graphs
• Given a graph G, its adjacency matrix A is $n \times n$ and
  – $a_{ij} = 1$, if there is a link from node i to node j
  – $a_{ij} = 0$, otherwise

\[
\begin{array}{c|c|c|c|c|c}
& v1 & v2 & v3 & v4 & v5 \\
v1 & 0 & 1 & 0 & 1 & 1 \\
v2 & 0 & 0 & 0 & 1 & 0 \\
v3 & 0 & 0 & 0 & 0 & 1 \\
v4 & 0 & 0 & 0 & 0 & 0 \\
v5 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]
PageRank: Matrix Notation

- $A \rightarrow$ Matrix containing the transition probabilities
  \[
  A = \varepsilon P^T + (1 - \varepsilon)E
  \]
- where $P_{ij} = 1/\text{out}(i)$, if there is a link from $i$ to $j$, 0 otherwise; $E$ is the random jumps matrix
- Probability distribution vector at time $k$
  \[
  \vec{x}^{(k)} = A^k \vec{x}^{(0)}
  \]
- $\vec{x}^{(0)}$ is the starting vector
- PageRank $\rightarrow$ Stationary distribution of the Markov Chain described by $A$, i.e., principal eigenvector or $A$
  \[
  \text{PageRank} = \lim_{k \to \infty} \vec{x}^{(k)}
  \]
PageRank in MapReduce

• Reconsider:  \[ PR(q) = \varepsilon \times \sum_{p|p \rightarrow q} \frac{PR(p)}{out(p)} + (1 - \varepsilon) \times \frac{1}{N} \]

⇒ to compute PR(q) we need only information about PR scores and out degree of nodes that link to q

Have info: (page q, PR) linking to page p1, p2, ...
⇒ Need to invert that pattern
PR in MR: Map Phase

• **map**\((\text{nid } m, \text{node } M)\)

\[ p = \frac{\text{M.pageRank}}{|\text{M.adjacencyList}|}\]

**emit**\((\text{nid } m, M)\)

**for all** \(\text{nid } x \text{ in } \text{M.adjacencyList}\) do

**emit**\((\text{nid } x, p)\)

- node has pageRank attribute and list of outgoing edges
- send info about outgoing edges
- “send” score mass to nodes M links to
PR in MR: After Map Phase

• We have now:

for page K (group by id of K):

[pageIN1, PR(IN1)/INn1],
[pageIN2, PR(IN2)/INn2],
...

[pageO1, pageO2, pageO3, ...]
PR in MR: Reduce Phase

- **reduce**\((\text{nid } m, [p_1,p_2,...])\)
  
  \[
s=0;\ M=\text{node}\\
  \text{for all } p \text{ in } [p_1,p_2,...] \text{ do}\\
  \quad \text{if } p \text{ instanceof node then}\\
  \quad \quad M = p\\
  \quad \text{else}\\
  \quad \quad s += p\\
  \quad M.\text{pageRank} = (1-\varepsilon)/N + \varepsilon\times s\\
  \quad \text{emit}(\text{nid } m, \text{ node } M)
  \]

  - recover outgoing edges
  - sum up incoming PR scores
Literature

- Jeffrey Dean und Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters“. Google Labs.
- Foto N. Afrati, Jeffrey D. Ullman: Optimizing joins in a map-reduce environment. EDBT 2010: 99-110
- Alper Okcan, Mirek Riedewald: Processing theta-joins using MapReduce. SIGMOD Conference 2011: 949-960
- Klaus Berberich, Srikanta J. Bedathur: Computing n-gram statistics in MapReduce. EDBT 2013: 101-112
- Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference 1993: 207-216
Literature (2)


