Algorithms in MapReduce

Krzysztof Dembczyński

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

Software Development Technologies
Master studies, second semester
Academic year 2017/18 (winter course)
Review of the previous lectures

- Mining of massive datasets.
- Classification and regression.
- Evolution of database systems
- MapReduce in Spark
  - The overall idea of the MapReduce paradigm.
  - WordCount and matrix-vector multiplication.
  - Spark: MapReduce in practice.
Outline

1 Motivation

2 Relational-Algebra Operations

3 Matrix Multiplication

4 Summary
Outline

1 Motivation

2 Relational-Algebra Operations

3 Matrix Multiplication

4 Summary
Algorithms in MapReduce

• How to implement fundamental algorithms in MapReduce?
  ▶ Relational-Algebra Operations.
  ▶ Matrix multiplication.
Outline

1 Motivation

2 Relational-Algebra Operations

3 Matrix Multiplication

4 Summary
Example (Relation **Links**)

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>url1</td>
<td>url2</td>
</tr>
<tr>
<td>url1</td>
<td>url3</td>
</tr>
<tr>
<td>url2</td>
<td>url3</td>
</tr>
<tr>
<td>url2</td>
<td>url4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Relational-Algebra Operations

- Selection
- Projection
- Union, Intersection, and Difference
- Natural Join
- Grouping and Aggregation
Relational-Algebra Operations

- $R, S$ - relation
- $t, t'$ - a tuple
- $C$ - a condition of selection
- $A, B, C$ - subset of attributes
- $a, b, c$ - attribute values for a given subset of attributes
Selection

- **Map:** For each tuple \( t \) in \( R \), test if it satisfies \( C \). If so, produce the key-value pair \((t, t)\). That is, both the key and value are \( t \).

- **Reduce:** The Reduce function is the identity. It simply passes each key-value pair to the output.
Selection

- **Map**: For each tuple $t$ in $R$, test if it satisfies $C$. If so, produce the key-value pair $(t, t)$. That is, both the key and value are $t$.

- **Reduce**: 
• **Map**: For each tuple \( t \) in \( R \), test if it satisfies \( C \). If so, produce the key-value pair \((t, t)\). That is, both the key and value are \( t \).

• **Reduce**: The Reduce function is the identity. It simply passes each key-value pair to the output.
Projection

• Map: For each tuple \( t \) in \( R \), construct a tuple \( t' \) by eliminating from \( t \) those components whose attributes are not in \( A \). Output the key-value pair \((t',t')\).

• Reduce: For each key \( t' \) produced by any of the Map tasks, there will be one or more key-value pairs \((t',t',...,t')\). The Reduce function turns \((t',[t',t',...,t'])\) into \((t',t')\), so it produces exactly one pair \((t',t')\) for this key \( t' \).
Projection

- **Map:**
  
  For each tuple $t$ in $R$, construct a tuple $t'$ by eliminating from $t$ those components whose attributes are not in $A$. Output the key-value pair $(t', t')$.

- **Reduce:**
  
  For each key $t'$ produced by any of the Map tasks, there will be one or more key-value pairs $(t', t',..., t')$. The Reduce function turns $(t', [t', t',..., t'])$ into $(t', t')$, so it produces exactly one pair $(t', t')$ for this key $t'$.  

Projection

• **Map**: For each tuple $t$ in $R$, construct a tuple $t'$ by eliminating from $t$ those components whose attributes are not in $A$. Output the key-value pair $(t', t')$.

• **Reduce**: 
Projection

- **Map**: For each tuple \(t\) in \(R\), construct a tuple \(t'\) by eliminating from \(t\) those components whose attributes are not in \(A\). Output the key-value pair \((t', t')\).

- **Reduce**: For each key \(t'\) produced by any of the Map tasks, there will be one or more key-value pairs \((t', t')\). The Reduce function turns \((t', [t', t', \ldots, t'])\) into \((t', t')\), so it produces exactly one pair \((t', t')\) for this key \(t'\).
• Map:

- Turn each input tuple \( t \) either from relation \( R \) or \( S \) into a key-value pair \( (t, t) \).

- Reduce: Associated with each key \( t \) there will be either one or two values. Produce output \( (t, t) \) in either case.
• **Map**: Turn each input tuple $t$ either from relation $R$ or $S$ into a key-value pair $(t, t)$.

• **Reduce**: 
• **Map**: Turn each input tuple $t$ either from relation $R$ or $S$ into a key-value pair $(t, t)$.

• **Reduce**: Associated with each key $t$ there will be either one or two values. Produce output $(t, t)$ in either case.
Intersection

- **Map:**

  - Turn each input tuple \( t \) either from relation \( R \) or \( S \) into a key-value pair \((t, t)\).
  - **Reduce:** If key \( t \) has value list \([t, t]\) then produce \((t, t)\). Otherwise, produce nothing.
Intersection

- **Map**: Turn each input tuple $t$ either from relation $R$ or $S$ into a key-value pair $(t, t)$.
- **Reduce**: 
Intersection

- **Map**: Turn each input tuple $t$ either from relation $R$ or $S$ into a key-value pair $(t, t)$.
- **Reduce**: If key $t$ has value list $[t, t]$, then produce $(t, t)$. Otherwise, produce nothing.
• **Map:**

  For a tuple \( t \) in \( R \), produce key-value pair \((t, \text{name}(R))\), and for a tuple \( t \) in \( S \), produce key-value pair \((t, \text{name}(S))\).

• **Reduce:**

  For each key \( t \), do the following.

  1. If the associated value list is \([\text{name}(R)]\), then produce \((t, t)\).

  2. If the associated value list is anything else, which could only be \([\text{name}(R), \text{name}(S)]\), \([\text{name}(S), \text{name}(R)]\), or \([\text{name}(S)]\), produce nothing.
• **Map**: For a tuple $t$ in $R$, produce key-value pair $(t, \text{name}(R))$, and for a tuple $t$ in $S$, produce key-value pair $(t, \text{name}(S))$.

• **Reduce**: 
• **Map**: For a tuple $t$ in $R$, produce key-value pair $(t, \text{name}(R))$, and for a tuple $t$ in $S$, produce key-value pair $(t, \text{name}(S))$.

• **Reduce**: For each key $t$, do the following.
  1. If the associated value list is $[\text{name}(R)]$, then produce $(t, t)$.
  2. If the associated value list is anything else, which could only be $[\text{name}(R), \text{name}(S)]$, $[\text{name}(S), \text{name}(R)]$, or $[\text{name}(S)]$, produce nothing.
Let us assume that we join relation $R(A, B)$ with relation $S(B, C)$ that share the same attribute $B$.

**Map:**
Natural Join

• Let us assume that we join relation $R(A, B)$ with relation $S(B, C)$ that share the same attribute $B$.

• **Map**: For each tuple $(a, b)$ of $R$, produce the key-value pair $(b, (\text{name}(R), a))$. For each tuple $(b, c)$ of $S$, produce the key-value pair $(b, (\text{name}(S), c))$.

• **Reduce**: 
Natural Join

- Let us assume that we join relation $R(A, B)$ with relation $S(B, C)$ that share the same attribute $B$.

- **Map**: For each tuple $(a, b)$ of $R$, produce the key-value pair $(b, (\text{name}(R), a))$. For each tuple $(b, c)$ of $S$, produce the key-value pair $(b, (\text{name}(S), c))$.

- **Reduce**: Each key value $b$ will be associated with a list of pairs that are either of the form $(\text{name}(R), a)$ or $(\text{name}(S), c)$. Construct all pairs consisting of one with first component $\text{name}(R)$ and the other with first component $S$, say $(\text{name}(R), a)$ and $(\text{name}(S), c)$. The output for key $b$ is $(b, [(a_1, b, c_1), (a_2, b, c_2), ...])$, that is, $b$ associated with the list of tuples that can be formed from an $R$-tuple and an $S$-tuple with a common $b$ value.
Grouping and Aggregation

• Let assume that we group a relation $R(A, B, C)$ by attributes $A$ and aggregate values of $B$.

• Map:

$$\text{For each tuple } (a, b, c) \text{ produce the key-value pair } (a, b).$$

• Reduce:

Each key $a$ represents a group. Apply the aggregation operator $\theta$ to the list $[b_1, b_2, ..., b_n]$ of $B$-values associated with key $a$. The output is the pair $(a, x)$, where $x$ is the result of applying $\theta$ to the list. For example, if $\theta$ is $\text{SUM}$, then $x = b_1 + b_2 + ... + b_n$, and if $\theta$ is $\text{MAX}$, then $x$ is the largest of $b_1, b_2, ..., b_n$. 
Grouping and Aggregation

- Let assume that we group a relation $R(A, B, C)$ by attributes $A$ and aggregate values of $B$.

- **Map**: For each tuple $(a, b, c)$ produce the key-value pair $(a, b)$.

- **Reduce**: 
Let assume that we group a relation $R(A, B, C)$ by attributes $A$ and aggregate values of $B$.

- **Map**: For each tuple $(a, b, c)$ produce the key-value pair $(a, b)$.
- **Reduce**: Each key $a$ represents a group. Apply the aggregation operator $\theta$ to the list $[b_1, b_2, \ldots, b_n]$ of $B$-values associated with key $a$. The output is the pair $(a, x)$, where $x$ is the result of applying $\theta$ to the list. For example, if $\theta$ is SUM, then $x = b_1 + b_2 + \ldots + b_n$, and if $\theta$ is MAX, then $x$ is the largest of $b_1, b_2, \ldots, b_n$. 
Outline

1 Motivation

2 Relational-Algebra Operations

3 Matrix Multiplication

4 Summary
Matrix Multiplication

- If $M$ is a matrix with element $m_{ij}$ in row $i$ and column $j$, and $N$ is a matrix with element $n_{jk}$ in row $j$ and column $k$, then the product:

$$P = MN$$

is the matrix $P$ with element $p_{ik}$ in row $i$ and column $k$, where:

$$p_{ik} =$$
Matrix Multiplication

- If $M$ is a matrix with element $m_{ij}$ in row $i$ and column $j$, and $N$ is a matrix with element $n_{jk}$ in row $j$ and column $k$, then the product:

$$P = MN$$

is the matrix $P$ with element $p_{ik}$ in row $i$ and column $k$, where:

$$p_{ik} = \sum_j m_{ij} n_{jk}$$
Matrix Multiplication

• We can think of a matrix $M$ and $N$ as a relation with three attributes: the row number, the column number, and the value in that row and column, i.e.,:

$$M(I, J, V) \text{ and } N(J, K, W)$$

with the following tuples, respectively:

$$(i, j, m_{ij}) \text{ and } (j, k, n_{jk}).$$

• In case of sparsity of $M$ and $N$, this relational representation is very efficient in terms of space.

• The product $MN$ is almost a natural join followed by grouping and aggregation.
Matrix Multiplication

• Map:
  Send each matrix element $m_{ij}$ to the key value pair: $(j, (M, i, m_{ij}))$.
  Analogously, send each matrix element $n_{jk}$ to the key value pair: $(j, (N, k, n_{jk}))$.

• Reduce:
  For each key $j$, examine its list of associated values. For each value that comes from $M$, say $(M, i, m_{ij})$, and each value that comes from $N$, say $(N, k, n_{jk})$, produce the tuple $(i, k, v = m_{ij}n_{jk})$.
  The output of the Reduce function is a key $j$ paired with the list of all the tuples of this form that we get from $j$: $(j, [(i_1, k_1, v_1), (i_2, k_2, v_2), ..., (i_p, k_p, v_p)])$. 

20 / 26
Matrix Multiplication

- **Map:**

  - **Reduce:** For each key $j$, examine its list of associated values. For each value that comes from $M$, say $(M,i,m_{ij})$, and each value that comes from $N$, say $(N,k,n_{jk})$, produce the tuple $(i,k,v = m_{ij}n_{jk})$.

  The output of the Reduce function is a key $j$ paired with the list of all the tuples of this form that we get from $j$: $(j,[[(i_1,k_1,v_1)],[(i_2,k_2,v_2)],...,[(i_p,k_p,v_p)]]).$
Matrix Multiplication

- **Map**: Send each matrix element $m_{ij}$ to the key value pair:

  $$(j, (M, i, m_{ij})).$$

  Analogously, send each matrix element $n_{jk}$ to the key value pair:

  $$(j, (N, k, n_{jk})).$$

- **Reduce**: 

  The output of the Reduce function is a key $j$ paired with the list of all the tuples of this form that we get from $j$:

  $$(j, [(i_1, k_1, v_1), (i_2, k_2, v_2), ..., (i_p, k_p, v_p)])$$.
Matrix Multiplication

- **Map**: Send each matrix element $m_{ij}$ to the key value pair:

  $$(j, (M, i, m_{ij})).$$

  Analogously, send each matrix element $n_{jk}$ to the key value pair:

  $$(j, (N, k, n_{jk})).$$

- **Reduce**: For each key $j$, examine its list of associated values. For each value that comes from $M$, say $(M, i, m_{ij})$, and each value that comes from $N$, say $(N, k, n_{jk})$, produce the tuple

  $$(i, k, v = m_{ij}n_{jk}),$$

  The output of the Reduce function is a key $j$ paired with the list of all the tuples of this form that we get from $j$:

  $$(j, [(i_1, k_1, v_1), (i_2, k_2, v_2), \ldots, (i_p, k_p, v_p)]).$$
Matrix Multiplication

Map:
From the pairs that are output from the previous Reduce function produce key-value pairs: 

\( (i_1, k_1), v_1 \), 
\( (i_2, k_2), v_2 \), ..., 
\( (i_p, k_p), v_p \).

Reduce:
For each key \((i, k)\), produce the sum of the list of values associated with this key. The result is a pair \((i, k), v\), where \(v\) is the value of the element in row \(i\) and column \(k\) of the matrix \(P = MN\).
Matrix Multiplication

• **Map:**

  - From the pairs that are output from the previous Reduce function produce key-value pairs: 
    
    \[
    ( (i_1, k_1), v_1), ( (i_2, k_2), v_2), \ldots, ( (i_p, k_p), v_p) \]

  - **Reduce:** For each key \((i, k)\), produce the sum of the list of values associated with this key. The result is a pair \(( (i, k), v)\), where \(v\) is the value of the element in row \(i\) and column \(k\) of the matrix \(P = MN\).
Matrix Multiplication

- **Map**: From the pairs that are output from the previous Reduce function produce $p$ key-value pairs:

  $$((i_1, k_1), v_1), ((i_2, k_2), v_2), \ldots, ((i_p, k_p), v_p).$$

- **Reduce**: 

  ...
Matrix Multiplication

- **Map**: From the pairs that are output from the previous Reduce function produce $p$ key-value pairs:

$$(((i_1, k_1), v_1), ((i_2, k_2), v_2), \ldots, ((i_p, k_p), v_p)).$$

- **Reduce**: For each key $(i, k)$, produce the sum of the list of values associated with this key. The result is a pair

$$((i, k), v),$$

where $v$ is the value of the element in row $i$ and column $k$ of the matrix

$$P = MN.$$
Matrix Multiplication with One Map-Reduce Step

- **Map:**

  For each element $m_{ij}$ of $M$, produce a key-value pair $((i,k), (M,j,m_{ij}))$, for $k = 1, 2, ..., \text{up to the number of columns of } N$.

  Also, for each element $n_{jk}$ of $N$, produce a key-value pair $((i,k), (N,j,n_{jk}))$, for $i = 1, 2, ..., \text{up to the number of rows of } M$. 
Matrix Multiplication with One Map-Reduce Step

- **Map**: For each element $m_{ij}$ of $M$, produce a key-value pair

$$((i, k), (M, j, m_{ij})),$$

for $k = 1, 2, \ldots$, up to the number of columns of $N$. Also, for each element $n_{jk}$ of $N$, produce a key-value pair

$$((i, k), (N, j, n_{jk})),$$

for $i = 1, 2, \ldots$, up to the number of rows of $M$. 
Matrix Multiplication with One Map-Reduce Step

- **Reduce:**

  Each key \((i, k)\) will have an associated list with all the values \((M,j,m)_{ij}\) and \((N,j,n)_{jk}\), for all possible values of \(j\). We connect the two values on the list that have the same value of \(j\):  
  
  ▶ We sort by \(j\) the values that begin with \(M\) and sort by \(j\) the values that begin with \(N\), in separate lists,  
  
  ▶ The \(j\)th values on each list must have their third components, \(m_{ij}\) and \(n_{jk}\) extracted and multiplied,  
  
  ▶ Then, these products are summed and the result is paired with \((i, k)\) in the output of the Reduce function.
Matrix Multiplication with One Map-Reduce Step

- **Reduce**: Each key \((i, k)\) will have an associated list with all the values

\[
(M, j, m_{ij}) \quad \text{and} \quad (N, j, n_{jk}),
\]

for all possible values of \(j\). We connect the two values on the list that have the same value of \(j\), for each \(j\):

- We sort by \(j\) the values that begin with \(M\) and sort by \(j\) the values that begin with \(N\), in separate lists,
- The \(j\)th values on each list must have their third components, \(m_{ij}\) and \(n_{jk}\) extracted and multiplied,
- Then, these products are summed and the result is paired with \((i, k)\) in the output of the Reduce function.
Outline

1. Motivation
2. Relational-Algebra Operations
3. Matrix Multiplication
4. Summary
Summary

- Algorithms in MapReduce:
  - Relational-algebra operations.
  - Matrix multiplication.
• A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets.*
  Cambridge University Press, 2011
  http://infolab.stanford.edu/~ullman/mmds.html

• J. Lin and Ch. Dyer. *Data-Intensive Text Processing with MapReduce.*
  Morgan and Claypool Publishers, 2010
  http://lintool.github.com/MapReduceAlgorithms/

• Ch. Lam. *Hadoop in Action.*
  Manning Publications Co., 2011