Nondeterminism in MapReduce
An Empirical Study on Non-commutative Aggregators in MapReduce Programs

Jiaqi Liu

October 23th 2015
1 Introduction
2 Commutativity
3 Conclusion
We will focus on **reduce function**
What if...
- mappers complete tasks in random order and
- reducers' output depends on order of input data (non-commutative)

Commutativity of reducer
A reducer is commutative if its output remains the same when its input rows are reordered

A non-commutative reducer can be a source of non-determinism
1. Introduction
2. Commutativity
3. Conclusion
Commutative reducers

Example

max(1, 2, 3) = max(3, 2, 1) = 3

Commutative operations

- SUM, COUNT, MAX, MIN, AND
- top ten, median
- sort then process
- ...

Reducer that performs operations above are commutative
Many reducers are non-commutative (58% of 507 in paper)

Programmers wrote them unintentially

5 categories

1. Single Item
2. Index Value Pair
3. Max Row
4. First N
5. Str Concat.
Non-commutative reducers
Single item - example

Example
row = (key, value1, value2)

Process input row in loop - select the first or last row

- for each row in input:
  x = row[1] // last row
- for each row in input:
  x = row[1]
  break // first row

- Reducer output depends on the value of x
- Input rows come in any order; x will get inconsistent first row and last row
Non-commutative reducers
Single item - how to eliminate

Functional Dependency (FD)
Column X functionally determines column Y (X --> Y) iff each X value is associated with exactly one Y value.

FD in MapReduce
If Key_reducer --> some value/column (e.g. row[1]), then we have deterministic results, since all row[1]'s must have the same value.

Example
user_ID --> user_gender
Non-commutative reducers

Index-value pair

Dictionary with 2 non-key values - $V_{\text{index}}: V_{\text{value}}$

<table>
<thead>
<tr>
<th>Input order</th>
<th>Input order</th>
</tr>
</thead>
<tbody>
<tr>
<td>key, 1, a</td>
<td>key, 1, a</td>
</tr>
<tr>
<td>key, 2, b</td>
<td>key, 2, c</td>
</tr>
<tr>
<td>key, 2, c</td>
<td>key, 2, b</td>
</tr>
<tr>
<td>Dictionary = {1: a, 2: b}</td>
<td>Dictionary = {1: a, 2: c}</td>
</tr>
</tbody>
</table>

How to ensure determinism

Again, functional dependency

$(\text{key}, V_{\text{index}}) \rightarrow V_{\text{value}}$
Non-commutative reducers
Max row

Find max ($V_{\text{max}}$), return other value ($V_{\text{other}}$)

- key, 1, a
- key, 2, b => emit (2, c)
- key, 2, c
- key, 1, a
- key, 2, c => emit (2, b)
- key, 2, b

How to ensure determinism (2 methods)

1. Again...functional dependency
   
   (key, $V_{\text{max}}$) --> $V_{\text{other}}$

2. Emit one with the largest timestamp (latest event)
   - In case of timestamp tie, pick any
Non-commutative reducers

First N

Return the first N rows and throw away the rest

Determinism?

- Maybe...if N is large
- Or you can tolerate nondeterminism
Non-commutative reducers

Str concat.

Concatenate strings of a column

Usage

- Object serialization for network communication

Determinism?

Depends on how succeeding jobs' logic

- Make a string set? Fine
- Print? No...
Outline

1 Introduction
2 Commutativity
3 Conclusion
Non-commutative reducer can be buggy in MapReduce program

Non-commutativity is not checked, for example in Hadoop

Programmers need to check manually:

1. Recognize non-commutativity pattern
2. Check data properties (FD, etc)
Thank you!