University of
South Australia

Design Patterns, Combiners and Dictionaries

1

What is a design pattern?

Resources for MapReduce Design Patterns are plentiful

Donald Miner and Adam Shook's book 'MapReduce Design Patterns' lists six types of patterns specifically for MapReduce

1. Summarization patterns

2

Pattern format in Miner & Shook

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

University of South Australia

3

Summarization patterns

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

University of South Australia

4

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Intent

Group records together by a key field and calculate a numerical aggregate per group

Consider f to be a generic numerical summarization function, given a list of values $(v_1, v_2, \ldots, v_n)$, we'd be trying to find a value α i.e. $\alpha = f(v_1, v_2, \ldots, v_n)$

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Motivation

Data sets are large these days! Too hard to discern meaning without a top-level view of the data.

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Applicability

Should be used when both of the following are true:

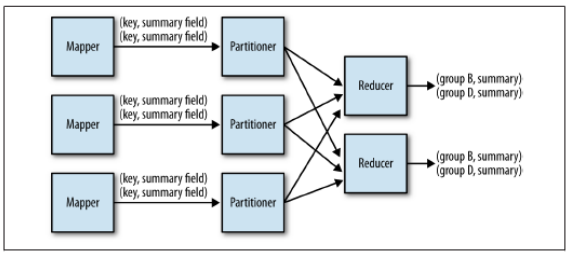
- You are dealing with numerical data
- The data can be grouped by specific fields

7

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Structure



MapReduce Design Patterns – Donald Miner & Adam Shook (2013)

8

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Consequences

The output of the job will be a set of files containing a single record per reducer input group. Each record will consist of the key and all aggregate values.

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Resemblances

Analogous patterns in both SQL and Pig

Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Known uses

Word count, Min/Max/Count, Average/Median/Standard Deviation

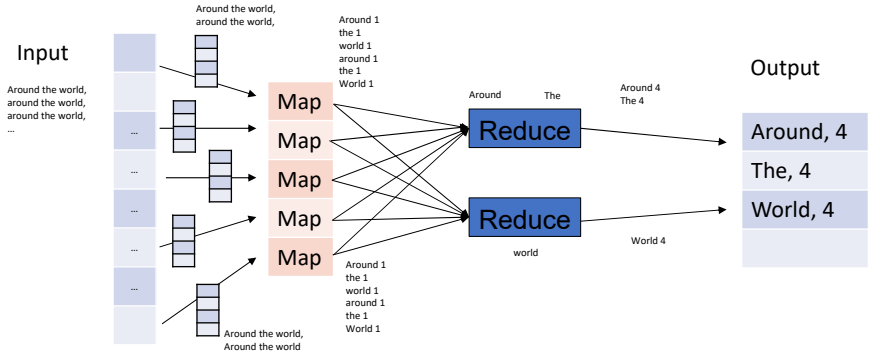
Numerical summarizations

- Intent
- Motivation
- Applicability
- Structure
- Consequences
- Resemblances
- Known uses
- Performance Analysis

Performance analysis

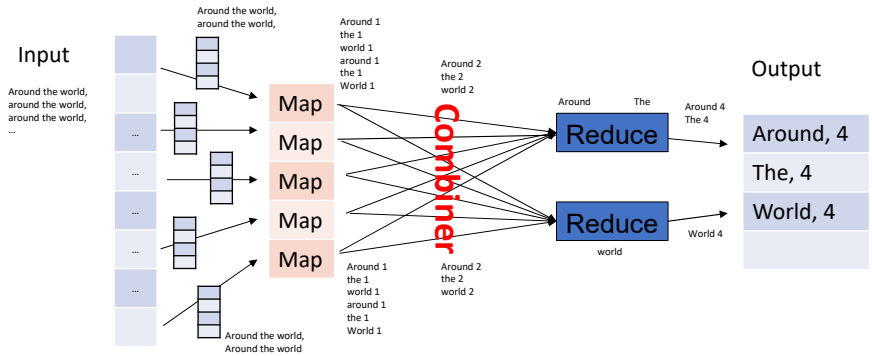
MapReduce was built for these types of jobs, it performs well. However there are still factors to be cautious of. The authors note that mastery of the combiner can significantly effect performance in some scenarios.

On the topic of combiners…



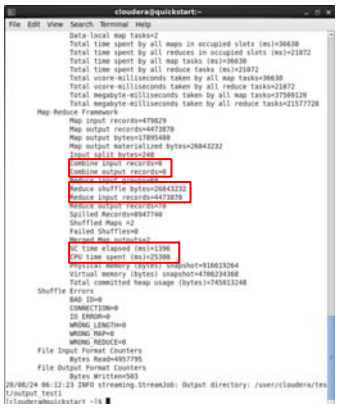
13

On the topic of combiners…



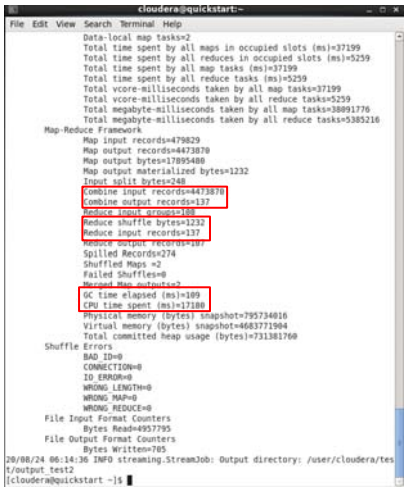
14

No combiner



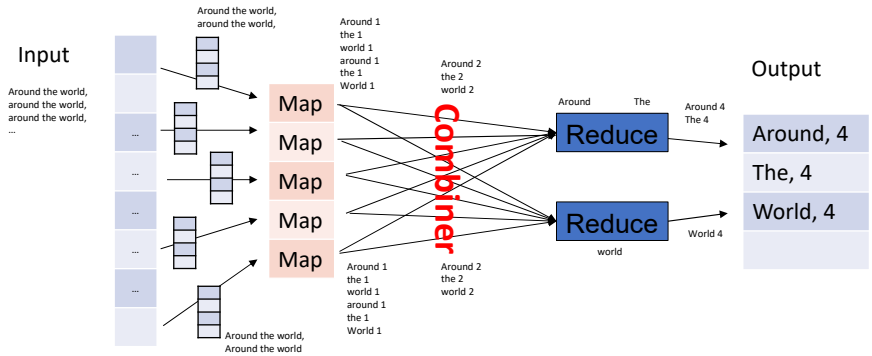
With combiner

```
[cloudera@quickstart ~]$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-mr1.jar -mapper /home/cloudera/test1/src/mapper.py -combiner /home/cloudera/test1/src/reducer.py -reducer /home/cloudera/test1/src/reducer.py -input /user/cloudera/test1/input/* -output /user/cloudera/test/output_test2
```



15

On the topic of combiners…



16

Dictionaries

In Python a dictionary is a collection which is unordered, changeable and indexed. Dictionaries are written with curly brackets, and they are also structured with key/value pairs

Define dictionary

```
myDictionary = {
"Name": "Mr Robot",
"Year": "2015",
"Seasons": "4"}
```

Access items

```
x = myDictionary["Year"]
or
x = myDictionary.get("Year")
```

Change items

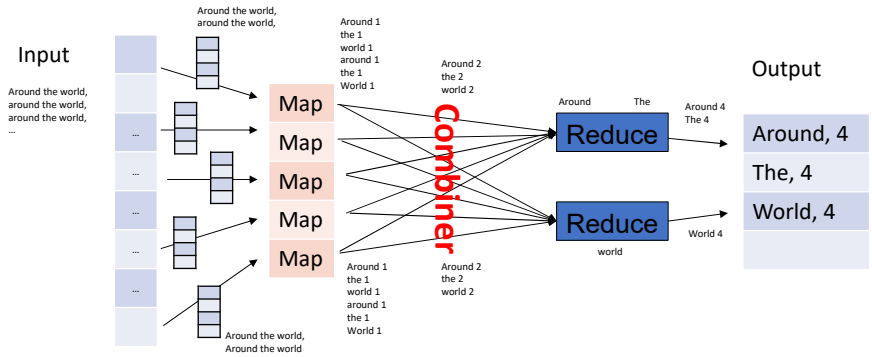
```
myDictionary["Year"] = 2019
```

Loop through keys and values

```
for x in myDictionary:
        print(x)
        print(myDictionary[x])
```

```
for x in myDictionary.values:
        print(x)
```


Combiners using dictionaries



lyrics = {} → "Around" IF → exists → lyrics["Around"] = lyrics["Around"]+1

IF → !exists → lyrics["Around"] = 1

→ Next word…