Concepts and Models of Parallel and Data-centric Programming

MapReduce Design Patterns – Filtering Patterns

Lecture, Summer 2020

Simon Schwitanski
Dr. Christian Terboven

Outline

Filtering Patterns

- Filtering patterns find subset of data (top-ten listing, deduplication, …)
- Understanding smaller piece of data
- Contrast to summarization patterns which provide top-level view of data
- Four different kinds of patterns
  - **Filtering**
  - Bloom Filtering
  - **Top Ten**
  - Distinct

Filtering

- Intent: Filter out records not of interest and keep other ones
  - Boolean evaluation function $f(r)$ taking record r as argument
  - Keep record r iff. $f(r)$ evaluates to *true*
- Motivation
  - Determining subsets out of a large dataset
  - Only process desired data in a follow-on analysis
- Applicability
  - Applicable on almost every kind of data
  - Only requirement: Data can be parsed into items that can be categorized by evaluation function (keep or drop)

Filtering – Structure

Pseudocode:

```
map(Object key, Object record):
    if keep_record(record) then
        Emit(key, record)
```

- Filtering only uses a Mapper, no Reducer
- Reason: No aggregation of data, just filtering

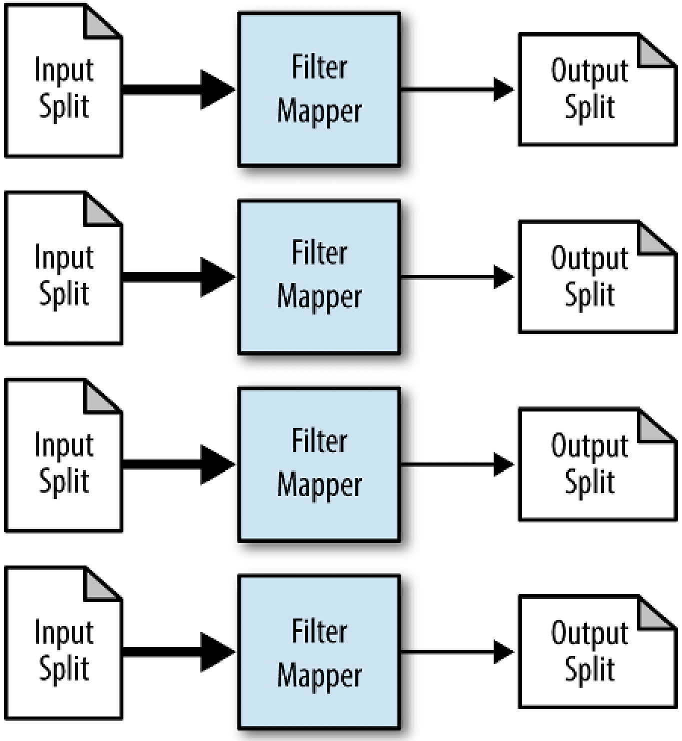

Illustration: Miner, Donald and Shook, Adam. "MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems", p.45, O'Reilly Media, 2012

Filtering – Applications

- Closer view of data: Analyze subset of data
- Distributed grep: Find lines of text that match a given regular expression
- Data cleansing: Clean up dirty data (malformed, incomplete, …)
- Random sampling: Randomly sample data from dataset (evaluation function randomly returns *true* or *false*)
- Note: Filtering similar to following SQL statement

```sql
SELECT * FROM mytable WHERE mypredicate;
```

where `mypredicate` represents a Boolean expression or function.

Filtering – Performance

- Map-only pattern, really efficient
  - No reducer, no data transmission needed
  - No sort phase, no reduce phase
  - Most Map tasks can pull data locally
- Note: Every map task will produce an output file, lots of files in case of high number of tasks.
- If a single or multiple large files desired: Use identity reducer with desired number of reduce tasks

Top Ten

- Intent: Retrieve small number of top K records using some ranking scheme
- Motivation:
  - Finding outliers which are typically most interesting
  - Avoid sorting the complete dataset, instead compute local sorts in mapper and merge them in reduce phase
- Applicability
  - Requires comparator function between two records
  - Number of output records should be “large enough”, otherwise total ordering of dataset can be simpler and faster
- Note: Getting top ten similar to following SQL statement

```sql
SELECT * FROM mytable ORDER BY mycol DESC LIMIT 10;
```

Top Ten – Structure

- Idea: Determine local top 10 of each input split in top ten mapper, determine final top 10 in a single top ten reducer.

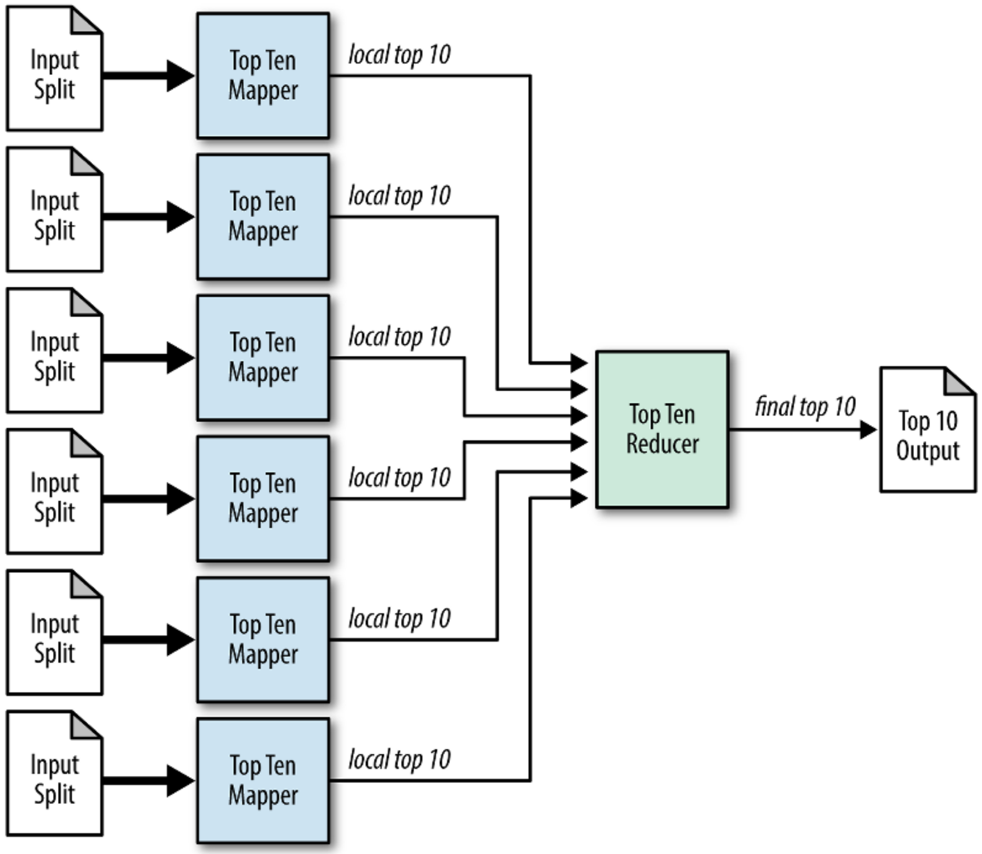

Illustration: Miner, Donald and Shook, Adam. “MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems”, p.60, O'Reilly Media, 2012

Top Ten – Performance

- Top ten just uses a single reducer, can lead to a bottleneck
  - Sort in the reducer can get expensive for huge number of entries
  - Network I/O concentrated on the host running the reducer
  - Writing map outputs to a single disk of the host running the reducer
- Large *K's* make pattern inefficient
  - Example: If the dataset has size 200,000 and *K* is 100,000 (larger than input split size), then each mapper sends out all his records to a single reducer. → Reducer has to handle all records.