CS 378 – Big Data Programming

Lecture 17

Filtering Patterns

Review

- Assignment 7 – User Sessions
  - Reduce side join (impression data, 2 sources)

- We'll look at implementation details of:
  - Avro schema
  - Populating Avro object with data
  - Mapper
  - Combiner
    - Should we use one? Can we use one?
  - Reducer

Filtering Patterns

- For filtering, we're not changing the data
- We interested in finding subsets of the data
  - Examine the data in detail
  - "Search"

- Sampling a common use of filtering
  - Create a representative subset for analysis

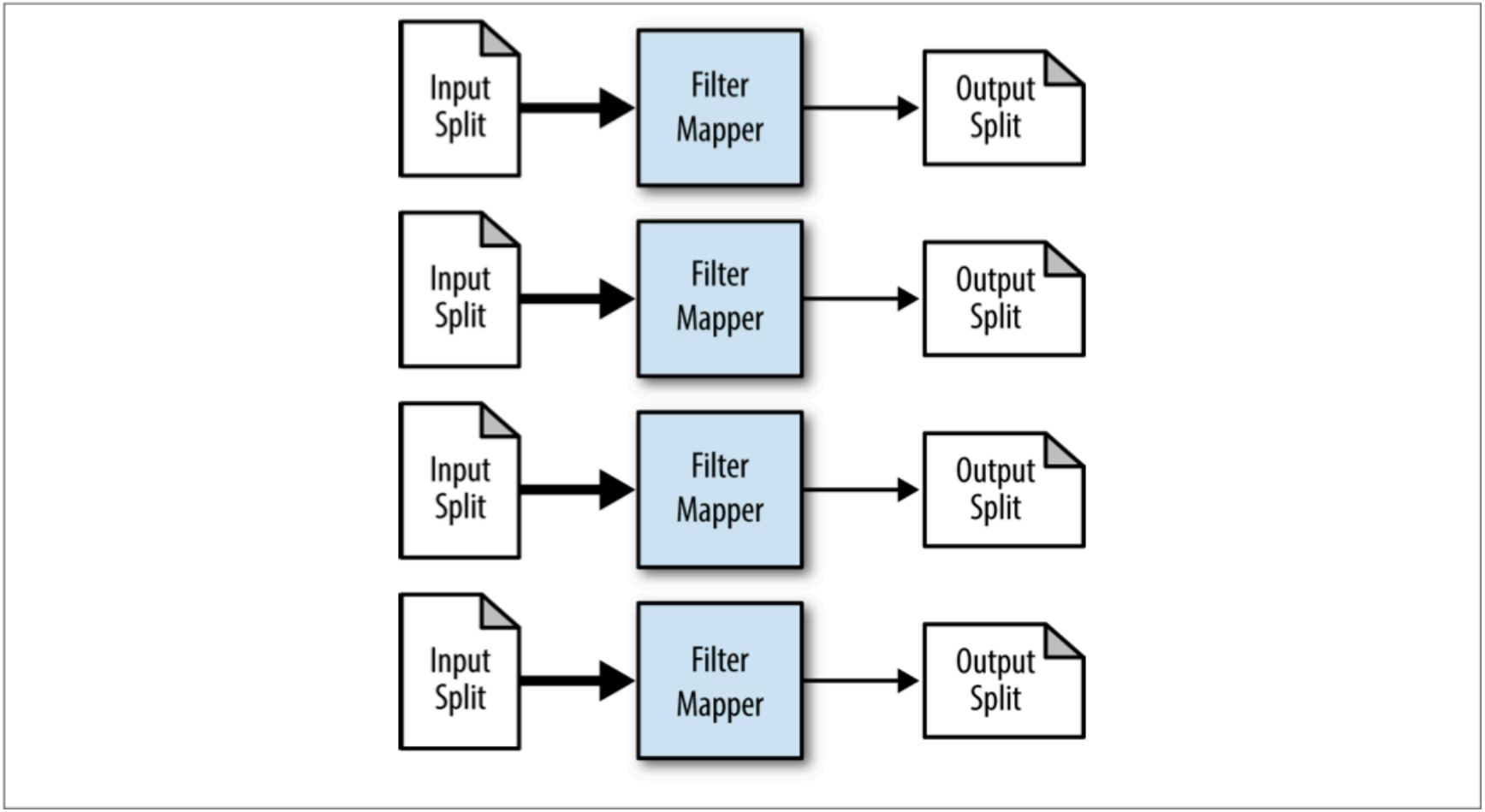
- Subset based on some relevance criteria

Filtering Patterns

- Basic Filtering
  - Examine each input record and decide whether it "stays"
- Apply a selection predicate to each input record
  - Return true if the record is to be kept (in the subset)
- MapReduce allows the filter to be applied in parallel
- Map-only

Basic Filtering- Data Flow

Figure 3-1 from MapReduce Design Patterns



Basic Filtering

- Map-only pattern
- Can we combine this with other patterns?
  - Other map-only patterns?
  - Patterns with reduce logic?
- Would we want to use `MultipleOutputs`?
- What sorts of filtering might we apply to sessions?

Basic Filtering

- Some common basic filtering uses

- grep
- Random sample
- Score records on some criterion, apply a threshold
- Data cleansing

Basic Filtering

- Since this is a map-only pattern, the number of output files will match the number of mappers
- If the filtering is strong, these files will be small

- What would we do to generate fewer, larger files?

- Use fewer mappers, but that would take longer
- Use identity mapper to consolidate output
  - Example of "chaining" jobs

Review - Multiple Outputs

- Hadoop class `MultipleOutputs`

- We saw this before with binning
  - Map-only pattern

- Since we have our user sessions completed in reduce
- Can we do the same thing (binning) in reduce output?
  - Suppose we want sessions to be "binned" or "partitioned" by some characteristic of the session

Assignment 8

- Consider the following categories of sessions:
- Levels of user engagement
  - "Submitter" – submitted a lead (or interacted with form)
    - Events: CHANGE, CONTACT_FORM_STATUS, EDIT, SUBMIT
  - "CPO" – not a Submitter session, has ILMR_CPO events
  - "Clicker" – not a CPO session, has click events
  - "Shower" – not a Clicker session, has show events
  - "Visitor" – not a Shower session, has visit events
  - "Other" – none of the above

- Patterns to implement
  - Filtering (sample certain sessions)
  - Binning (use `AvroMultipleOutputs`, map only job)

Multiple Outputs Setup

- In the `run()` method, specify the named output

```
MultipleOutputs.addNamedOutput(job, outputName,
    TextOutputFormat.class, Text.class, Text.class);
```

- For AVRO output

```
AvroMultipleOutputs.addNamedOutput(job, sessionType,
    AvroKeyValueOutputFormat.class,
    key schema, value schema);
```

- Enable counters for the multiple outputs

```
AvroMultipleOutputs.setCountersEnabled(job, true);
```

Multiple Outputs Setup

- In the map class, define an instance variable

```
private AvroMultipleOutputs multipleOutputs;
```

- In the `setup()` method of the map class

```
public void setup(Context context) {
    multipleOutputs = new AvroMultipleOutputs(context);
}
```

- In `map()` method (AVRO or text):

```
multipleOutputs.write(sessionType, key, value);
```

- In the `cleanup()` method of the map class

```
public void cleanup(Context context)
        throws InterruptedException, IOException{
    multipleOutputs.close();
}
```

Assignment 8

- Notes:

- Input is an AVRO container file (`dataSet8.avro`) on S3
- To read this file use the `sessions.avsc` provided
  - See: Canvas / Files / Assignment 8

- Use the enum `SessionType.java` provided
  - See: Canvas / Files / Assignment 8
  - Use `getText()` for the name to use as "namedOutput"