CS 378 – Big Data Programming

Lecture 14

more on

Data Organization Patterns

Review

- Assignment 6 – User Sessions

- Questions/Issues?
  - Selecting default values
    - Lat/lon
  - Hadoop reusing the same instance
    - From the `Iterable` to `reduce()`
  - Fields with a default of null

  - Determining whether a field is a "session" field
    - Has the same value for all impressions with the same userID/apikey

Assignment 6

- Define an Avro object for user session
  - One user session for each unique userID/apikey
  - Session will include an array of impressions
  - Impressions ordered by timestamp
  - Each impression will contain an array of IDs (0 or more)

- Identify data associated with the session as a whole
- Identify data associated with individual impressions
- Include all the fields listed in the assignment
- Create enums where requested

Data Organization Patterns

- We’ve discussed:
- Structured to hierarchical pattern
- Partitioning

- Let’s look at:
- Binning
- Shuffle
- Total Order Sorting

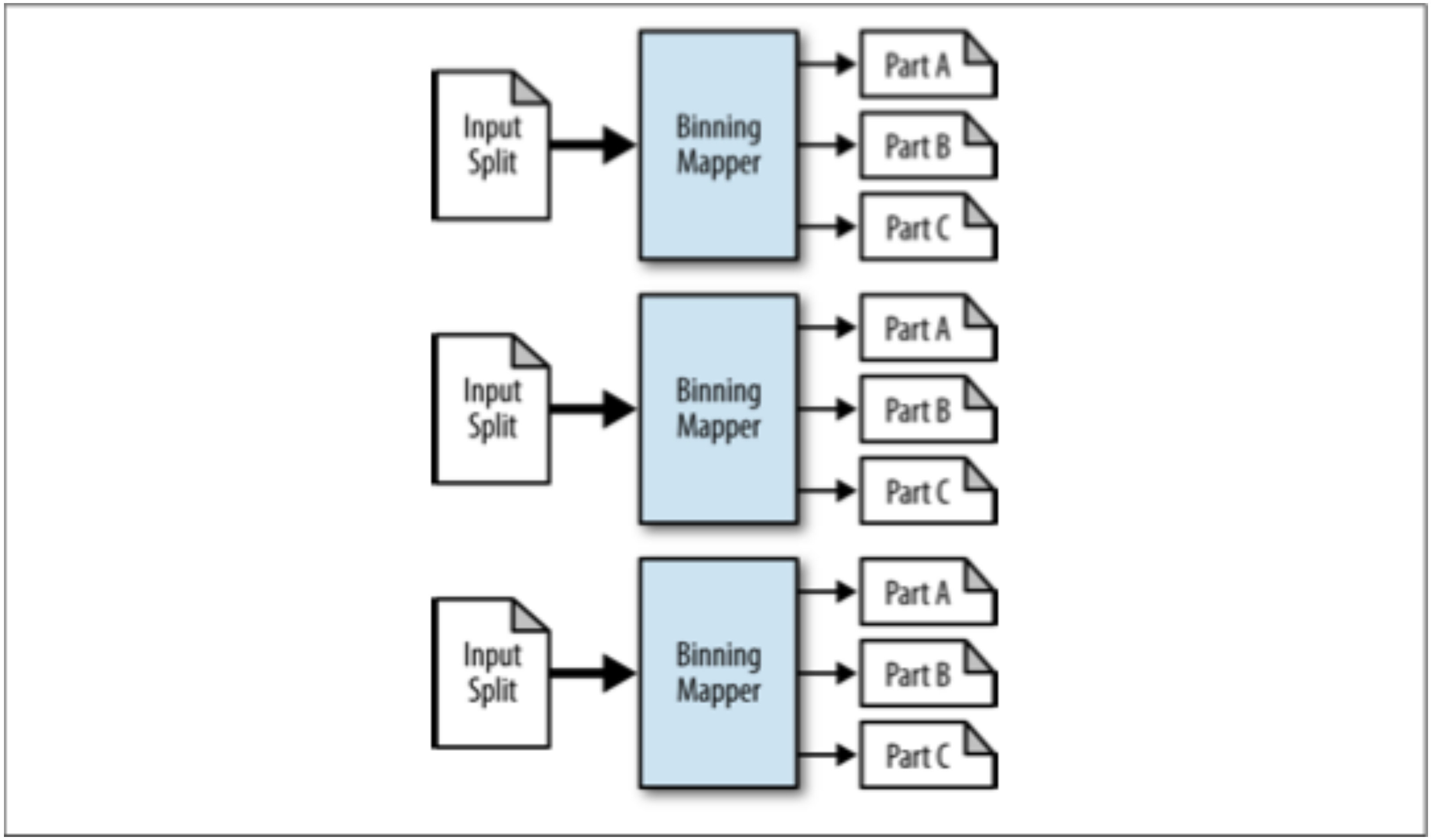
Binning

- Similar to partitioning
  - Want to organize output into categories
  - Map-only pattern (# reduce tasks set to 0)

- Mapper output written to output directories
- Uses `MultipleOutputs` class
  - Call `write()` on `MultipleOutputs`, not `Context`
  - For each category, each mapper writes a file
  - Expensive if many mappers and many categories

Binnig Data Flow

Figure 4-3 from MapReduce Design Patterns



Shuffle

- Want to distribute output randomly
- Mapper generates a random key for each output
- If you want to reuse a mapper, you could add a partitioner that generates a random partition #
  - Mapper code is then unchanged
- Reducer can sort based on some other random key
  - Further shuffling the data (input order now gone)

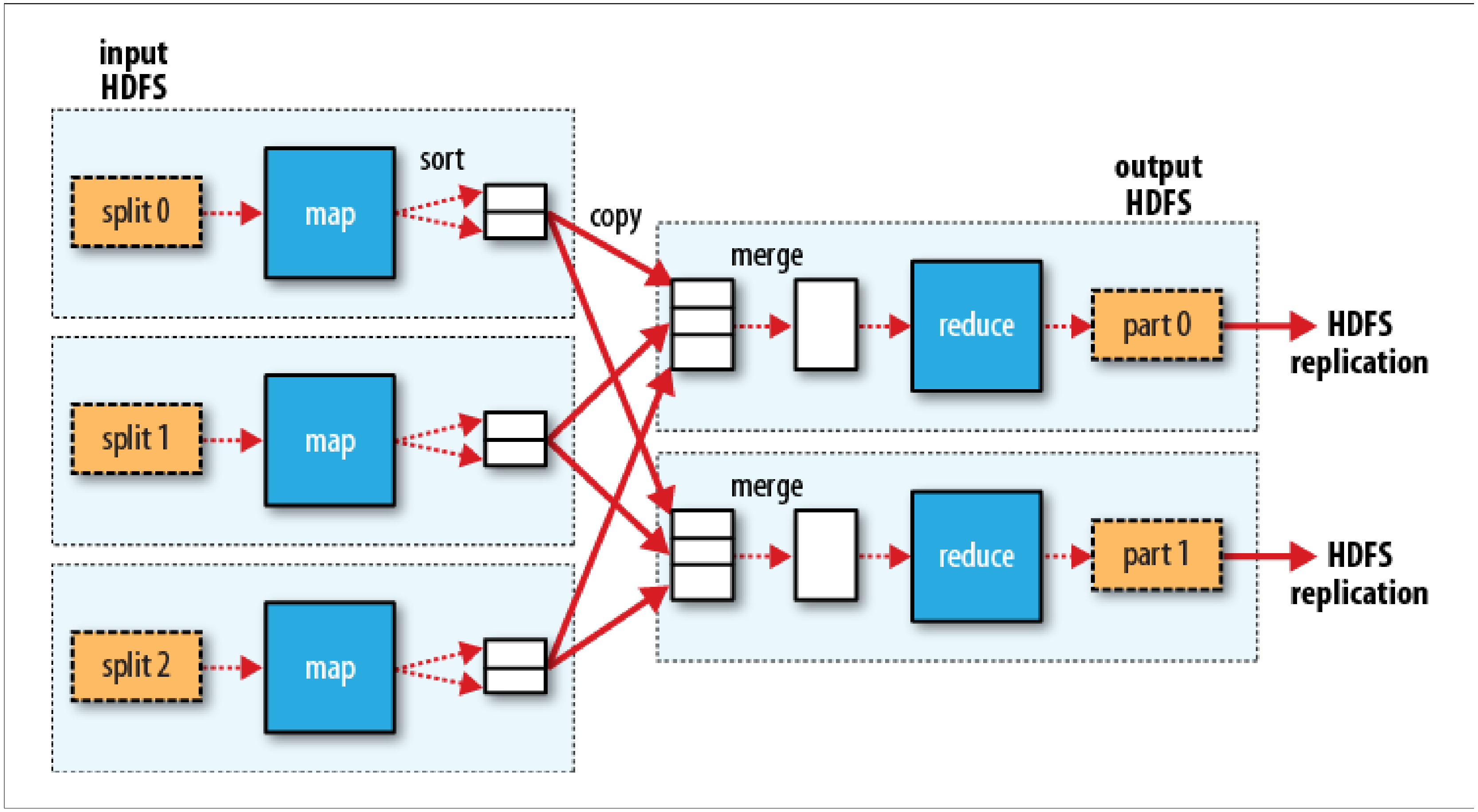
Shuffle – Why Do This?

- Random sampling
- Randomly select subset of the data (downsample)
- Multiple random subsets for
  - Model generation and testing – cross validation
  - Train on 80%, test on 20%, for 5-fold cross validation

- Anonymizing data (example from the textbook)
  - Replace PII with a random key

MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



Total Order Sorting

- Individual reducers can sort their keys
  - Need to retain an data in memory
  - Not sorted when concatenated with other reducer output

- We can identify subranges of the key space
  - We know the sort position of each subrange relative to other subranges
  - Use a partitioner to assign a key to its subrange
  - Reducer simply outputs the values. Why?

Total Order Sorting

- Issues in selecting subranges of the key space
- Would like subranges to be roughly equivalent in size
  - Can do an analysis of the key space by random sample
  - Will be a separate mapReduce job
  - Need to redo this analysis if key distribution changes
- Subranges ideas for our session key space?

Total Order Sorting

- Hadoop provides `TotalOrderPartitioner`
- Have to provide a “partition file”
  - Specifies the key range of each partition
  - Number of reducers must equal number of partitions
- Custom partitioner for our user session key space
  - Based on userId/apikey
  - Other data to use for sort?