CS 378 – Big Data Programming

Lecture 11

more on

Data Organization Patterns

Assignment 5 - Review

- Define an Avro object for user session
  - One user session for each unique userID
  - Session will include an array of events
  - Events ordered by timestamp

- Identify data associated with the session as a whole
- Identify data associated with individual events
- Include all the fields in the log entries
- Create enums where requested

Data Organization Patterns

- Structured to hierarchical pattern
  - User session is one such example
    - Organizing web logs by user
  - Textbook shows organizing posts and comments from StackOverflow

Partitioning

- Organize “similar” records into partitions

- Why?
  - Future jobs will only focus on subsets of the data
- Partitioning schemes:
  - Time: hour, day, week, month, year
  - Geography: ZIP, DMA, state, time zone, country
  - Data source: web site
  - Data type

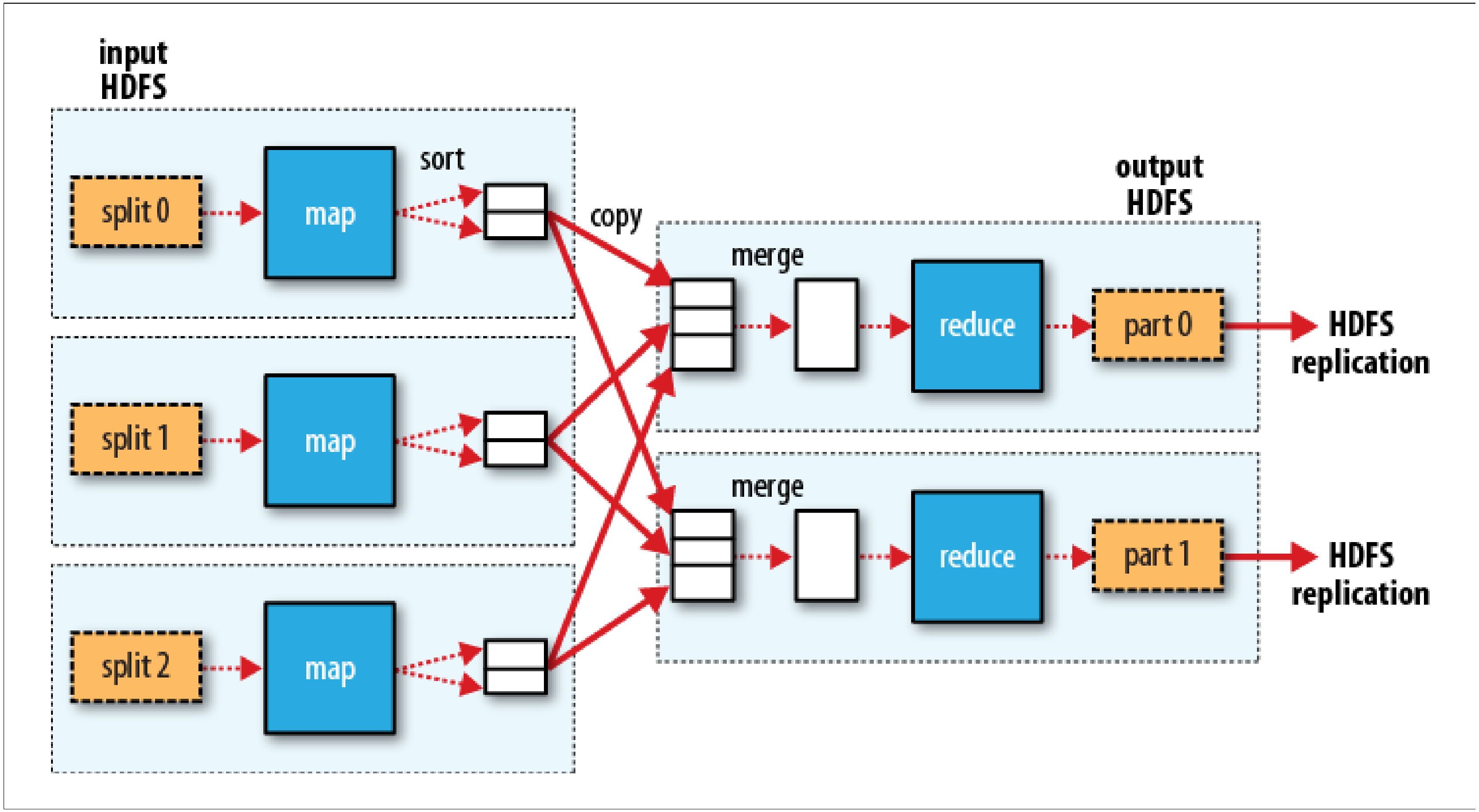
Partitioning

- No downside, as a mapReduce job can run over all partitions if needed
  - Note: Avoid creating many small files

- We do need to know *a priori* how many partitions we want
  - Can run a job that scans and summarizes the data
  - Get possible values, and counts
  - Just like we did for user sessions

MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



Partitioning

- What are some of the ways we might partition our user sessions?

- How would we do this?

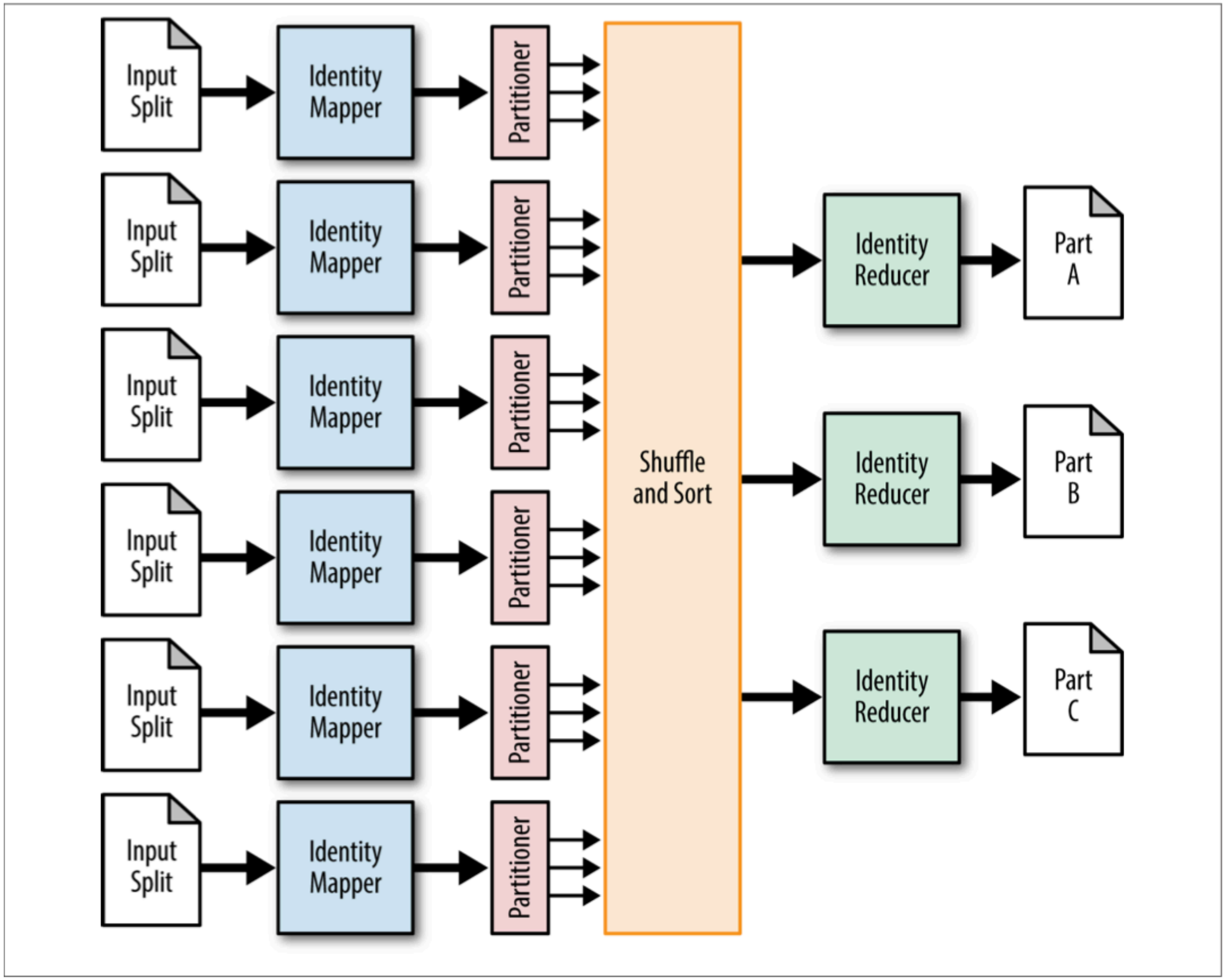
Partitioning

- Define a `Partitioner`
- Examines each `map()` output pair
- Computes a partition number

- Example

Data Flow

Figure 4-2 from MapReduce Design Patterns



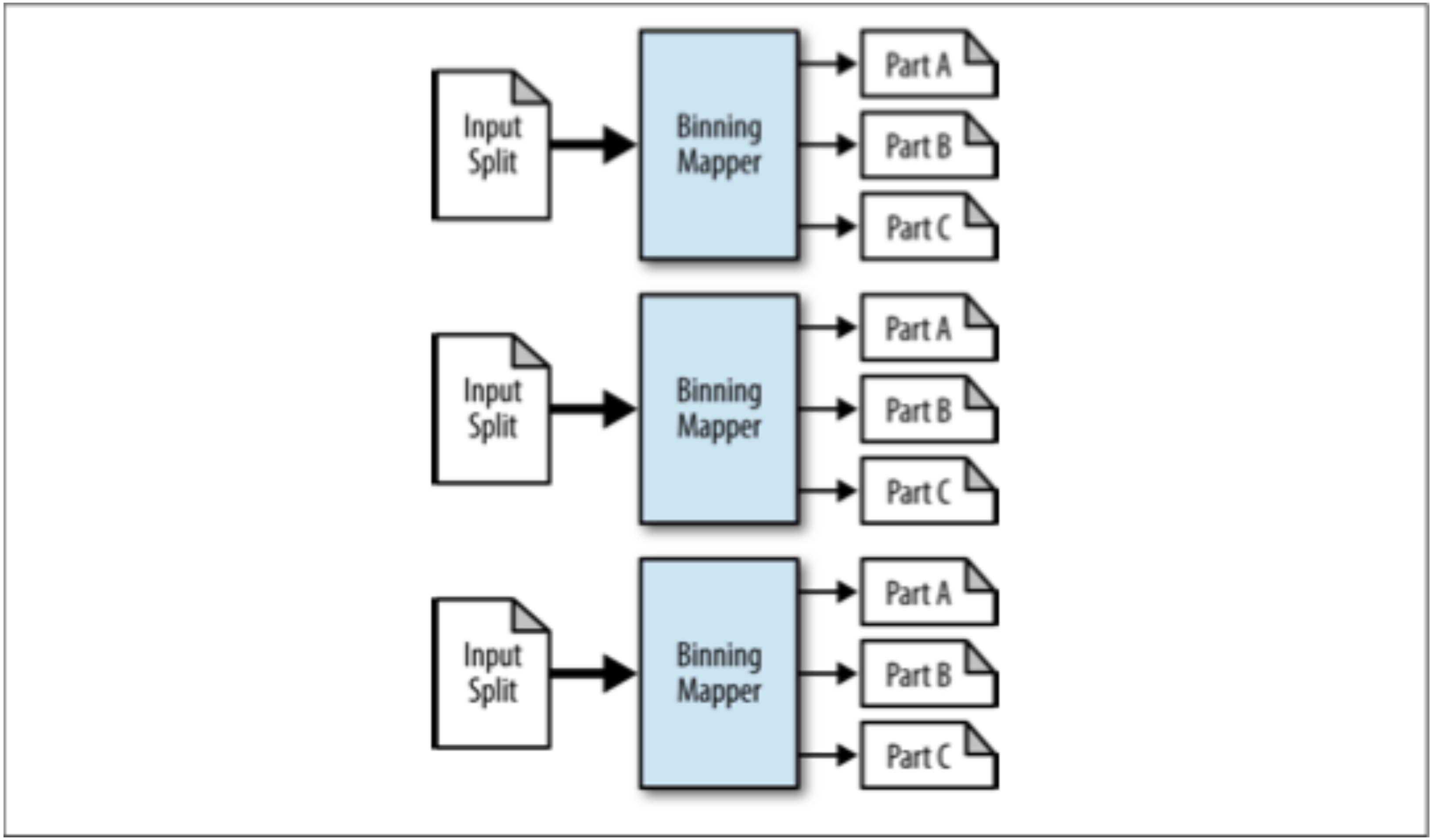
Binning

- 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

Binning Data Flow

Figure 4-3 from MapReduce Design Patterns



Discussion

- When should we use partitioning?
- When should we use binning?
- Consider the user sessions we’ve created

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

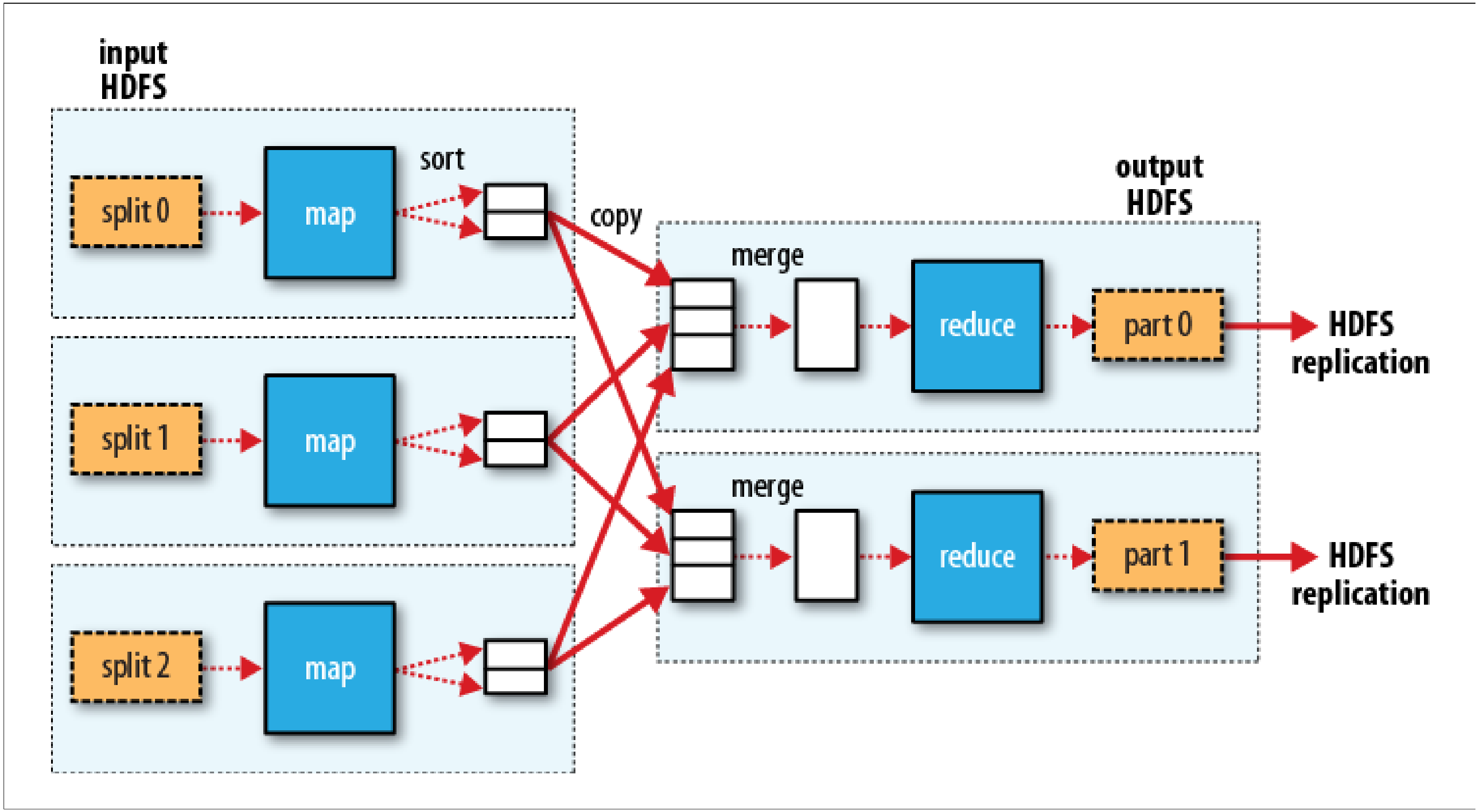
- Anonymizing data (example from the textbook)
  - Replace PII with a random key

Discussion

- Suppose we wanted to shuffle our user sessions
- When could we accomplish this in the process of creating sessions?

MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



Total Order Sorting

- Individual reducers can sort their keys
  - Need to retain all 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

- Subrange 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
  - Other data to use for sort?