MapReduce

Introduction to Information Retrieval
INF 141/ CS 121
Donald J. Patterson

Content adapted from Hinrich Schütze
http://www.informationretrieval.org

Distributed Indexing - Architecture

- 'MapReduce' is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster.
- Computational processing can occur on data stored either in a filesystem (unstructured) or in a database (structured).
- MapReduce can take advantage of locality of data, processing data on or near the storage assets to decrease transmission of data.

wikipedia

Distributed Indexing - Architecture

- "Map" step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes.
- The worker node processes the smaller problem, and passes the answer back to its master node.
- "Reduce" step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve.

wikipedia

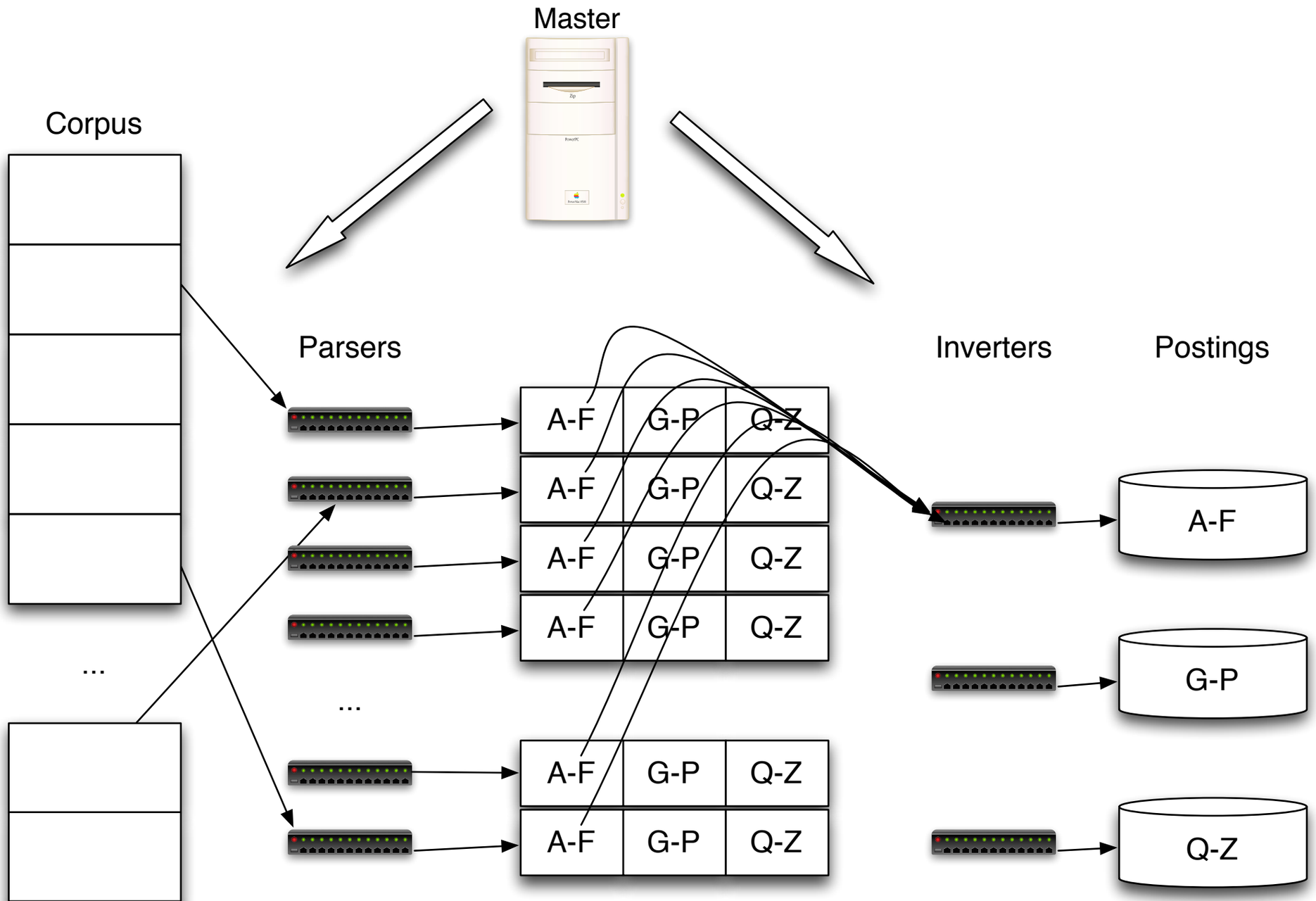
Distributed Indexing - Architecture

- Generally speaking in MapReduce
  - There is a map phase
    - This takes input and makes key-value pairs
    - this corresponds to the “parse” phase of BSBI and SPIMI
  - The map phase writes intermediate files
    - Results are bucketed into R buckets
- There is a reduce phase
  - This is the “invert” phase of BSBI and SPIMI
  - There are R inverters

Distributed Indexing - Architecture

- Use an instance of MapReduce
  - A general architecture for distributed computing jobs
  - Manages interactions among clusters of
    - cheap commodity compute servers
    - aka nodes
  - Uses Key-Value pairs as primary object of computation
  - An open-source implementation is “Hadoop” by apache.org

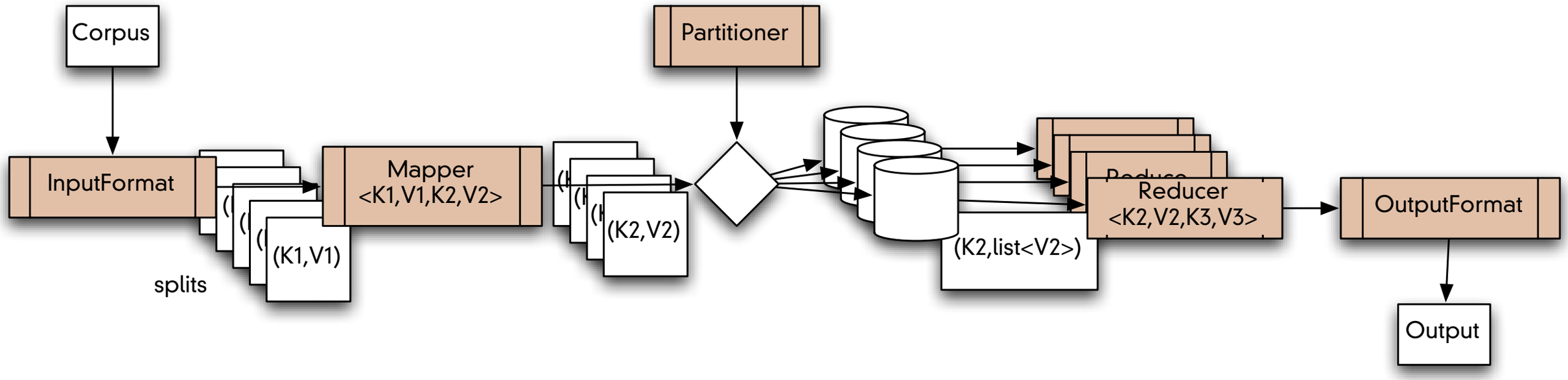
Distributed Indexing - Architecture



Distributed Indexing - Architecture

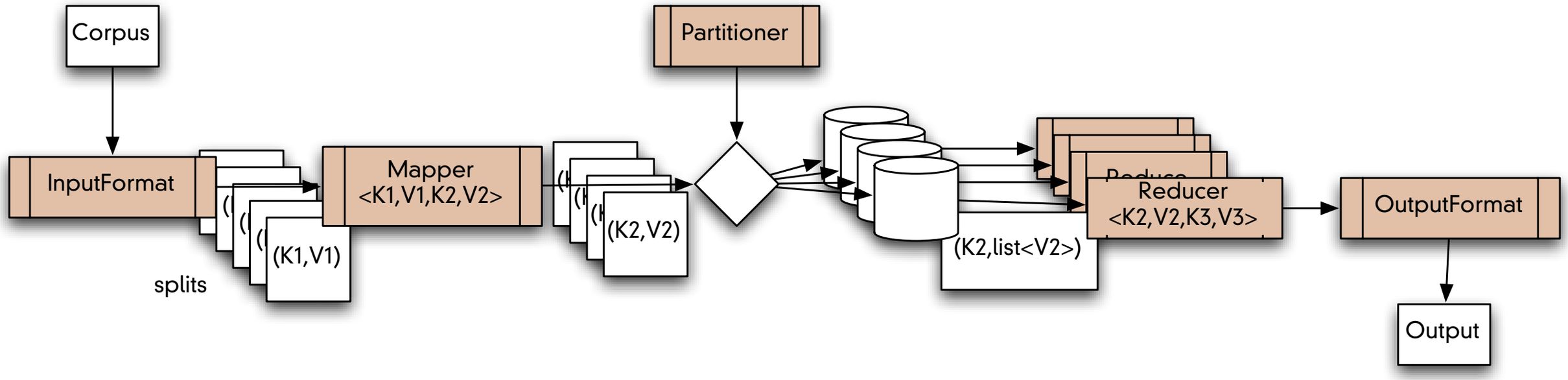
- Parsers and Inverters are not separate machines
  - They are both assigned from a pool
  - It is different code that gets executed
- Intermediate files are stored on a local disk
  - For efficiency
  - Part of the “invert” task is to talk to the parser machine and get the data.

Distributed Indexing - Hadoop



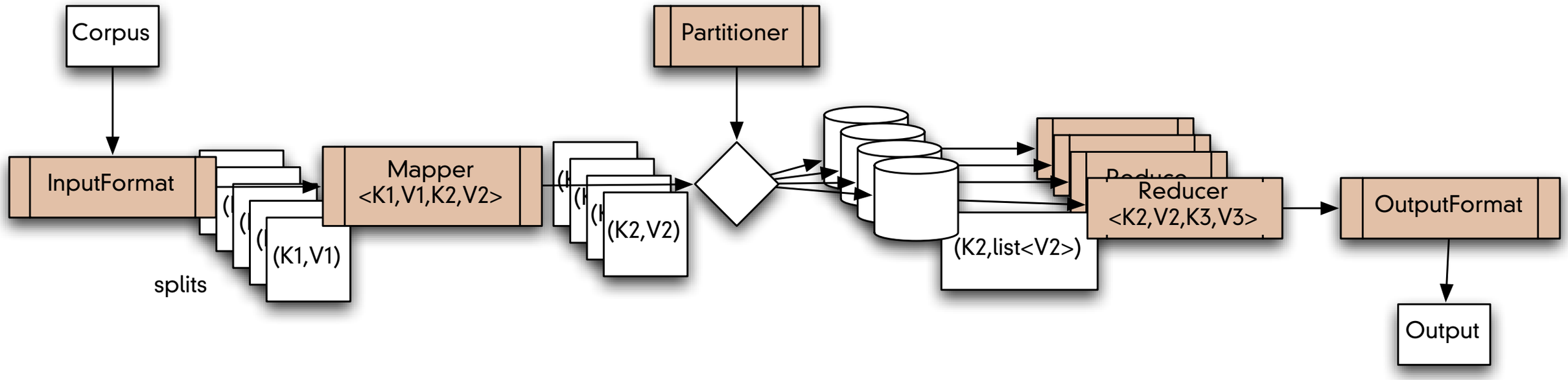
- InputFormat
  - Creates splits
  - One split is assigned to one mapper
  - A split is a collection of <K1,V1> pairs

Distributed Indexing - Hadoop



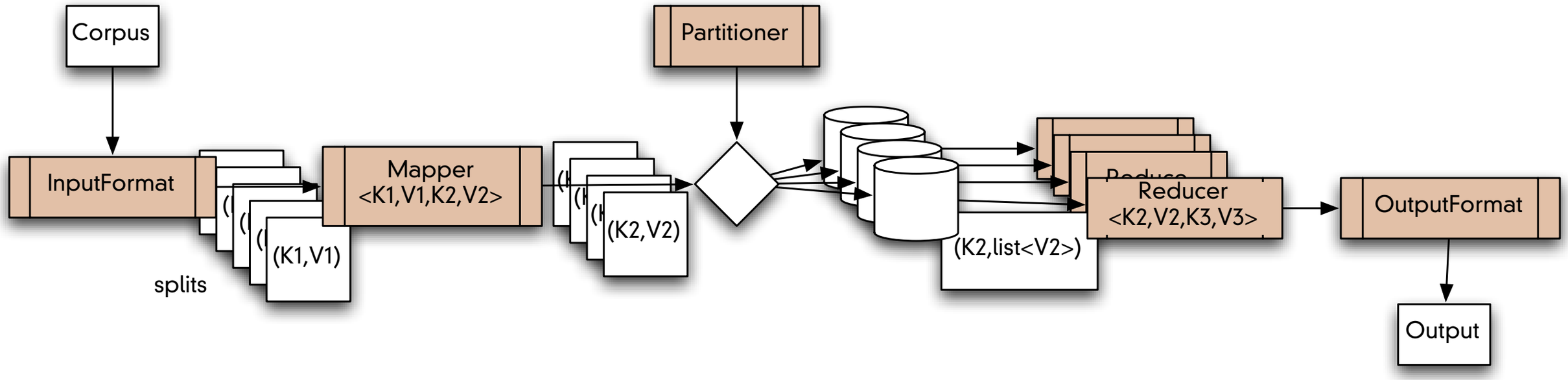
- InputFormat
  - Hadoop comes with NLineInputFormat which breaks text input into splits with N lines each
  - K1 = line number
  - V1 = text of line

Distributed Indexing - Hadoop



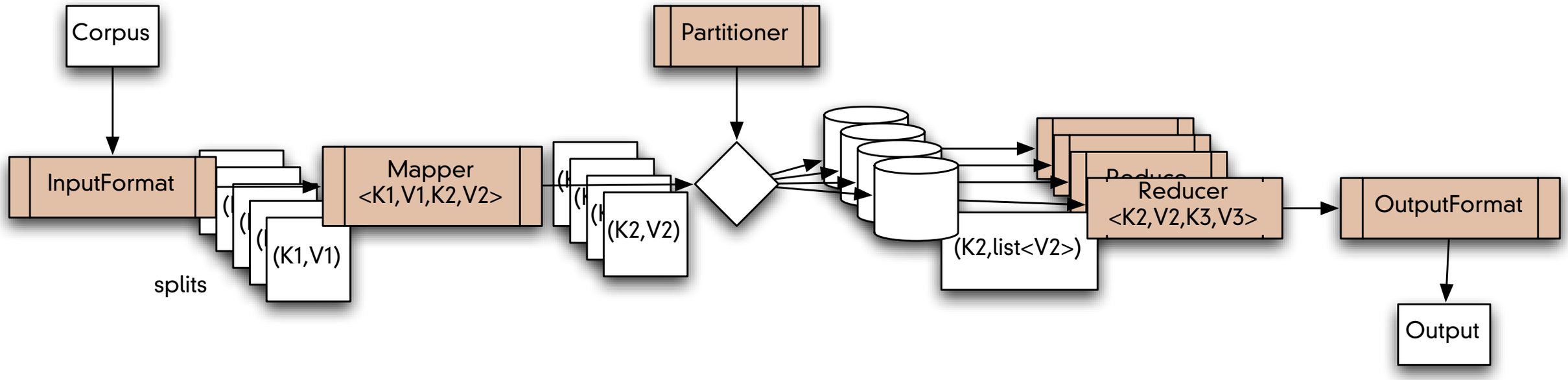
- Mapper<K1,V1,K2,V2>
  - Takes a <K1,V1> pair as input
  - Produces 0, 1 or more <K2,V2> pairs as output
  - Optionally it can report progress with a Reporter

Distributed Indexing - Hadoop



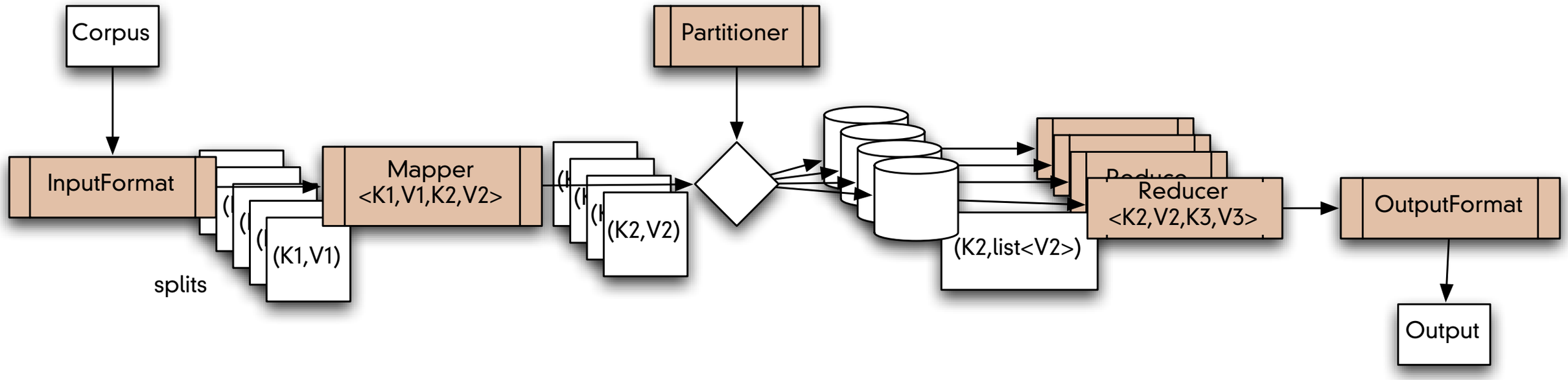
- Partitioner<K2,V2>
  - Takes a <K2,V2> pair as input
  - Produces a bucket number as output
  - Default is HashPartitioner

Distributed Indexing - Hadoop



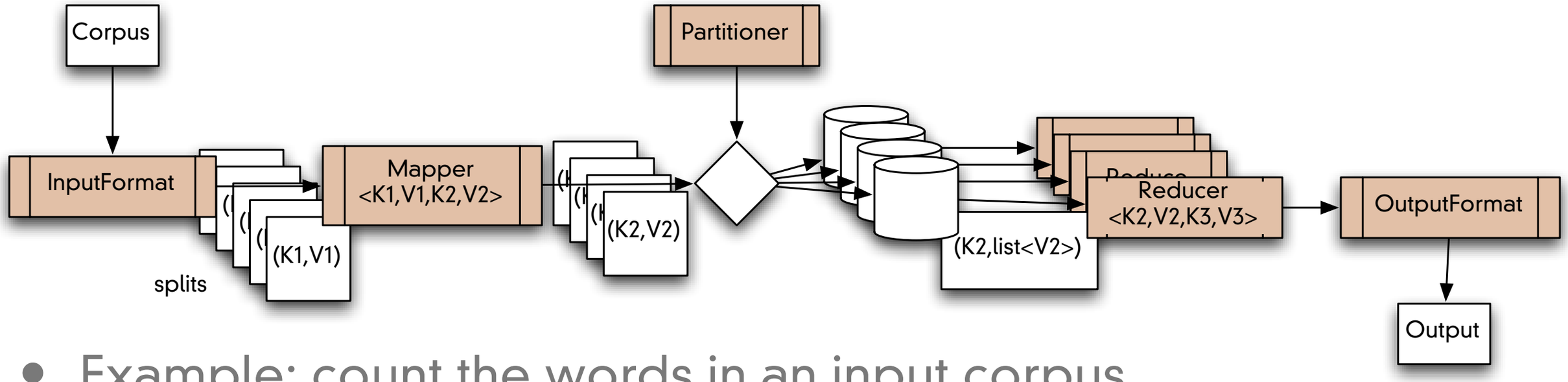
- Reducer<K2,V2,K3,V3>
  - Takes a <K2,list<V2>> pair as input
  - Produces <K3,V3> as output
  - Output is not resorted

Distributed Indexing - Hadoop



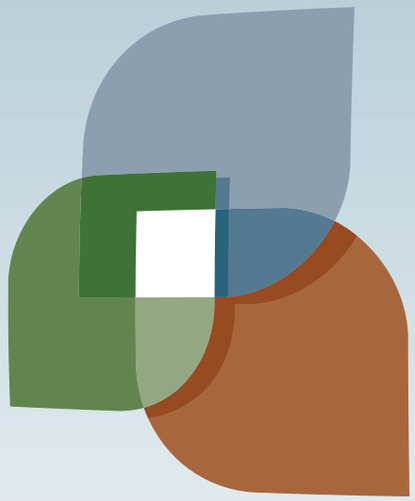
- OutputFormat
  - Does something with the output (like write it to disk)
  - TextOutputFormat<K3,V3> comes with Hadoop

Hadoop example: WordCount



- Example: count the words in an input corpus
  - InputFormat = TextInputFormat
  - Mapper: separates words, outputs <Word, 1>
  - Partitioner = HashPartitioner
  - Reducer: counts the length of list<V2>, outputs <Word,count>
  - OutputFormat = TextOutputFormat

End of Chapter 4



LUCI