Parallel & Distributed Computing (WQD7008)

Week 11

Workflow for Data Intensive Applications Management

2019/2020 Semester 1

Dr. Hamid Tahaei

Introduction

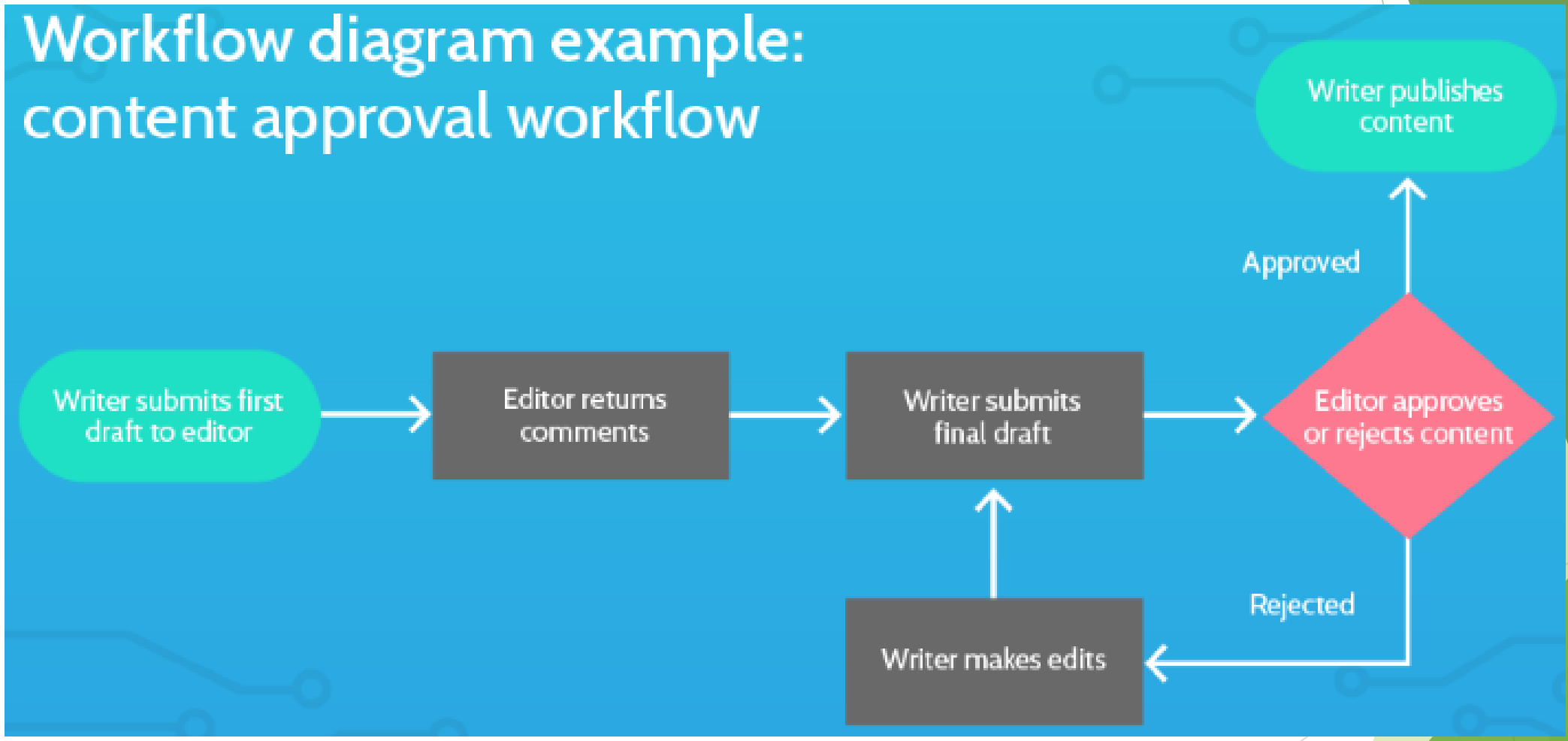
What Data Intensive Application?

- Many applications today are *data-intensive*, as opposed to *compute-intensive*.
- Data-intensive = data is its primary challenge
  - Quantity of data
  - Complexity of data
  - Speed of change
- Opposed to compute-intensive where CPU cycles are the bottleneck.
- Commonly needed functionality: Databases, Caches, search indexes, stream processing, batch processing.

What is workflow?

- A workflow consists of
  - an orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes.
  - Transform materials, provide services, or process information It can be depicted as a sequence of operations, declared as work of a person or group, an organization of staff, or one or more simple or complex mechanisms.
- Workflows are the way people get work done, and can be illustrated as series of steps that need to be completed sequentially in a diagram or checklist.

Introduction

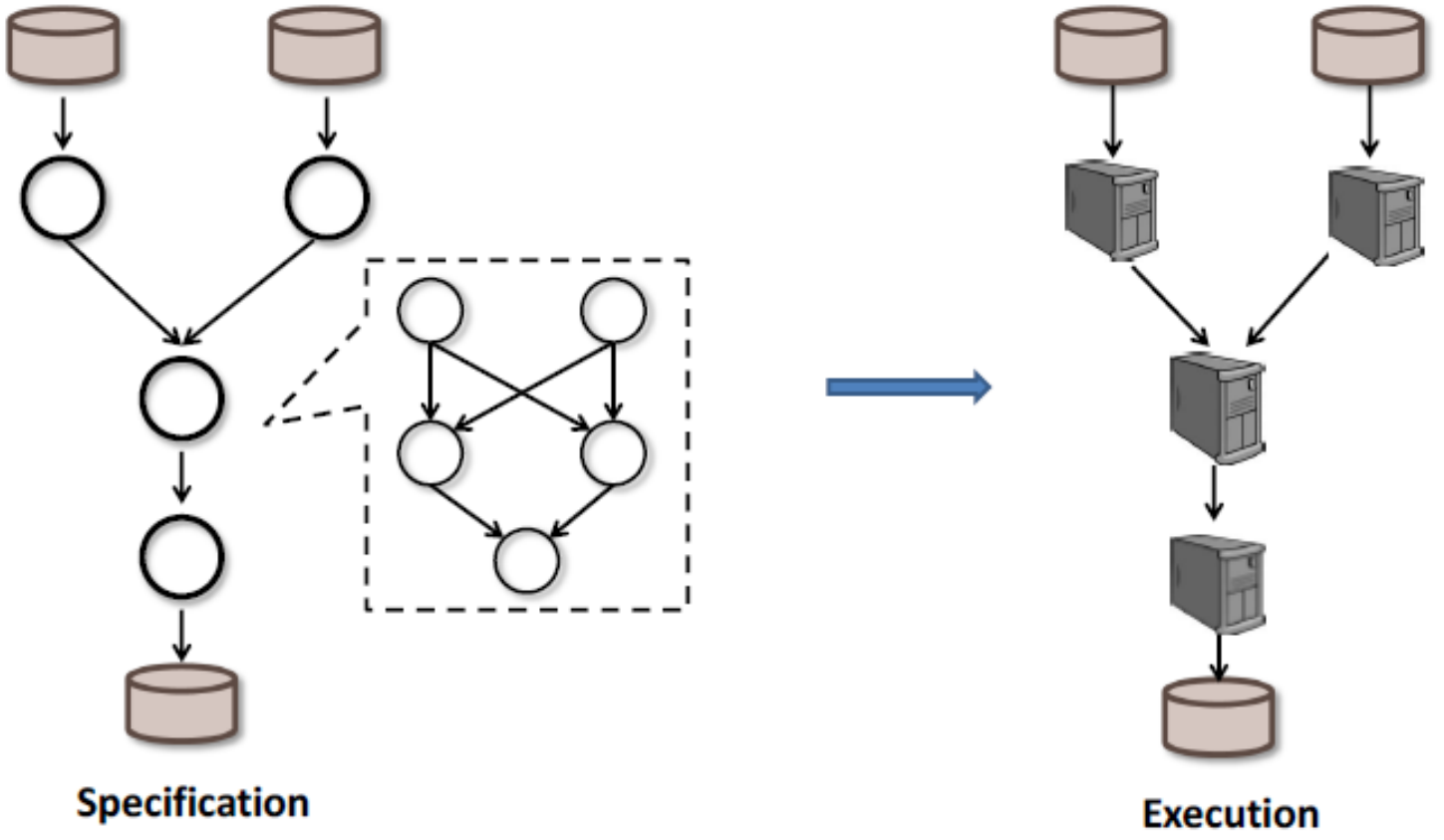


Overview of Data Processing Workflow

- The data processing workflow is often used to represent the execution of complex data-intensive analytics.
- Typically consists of a sequence of inter dependent data processing operators:
  - data loading,
  - information filtering,
  - data transformation & analysis and
  - other user-defined data manipulation.

Overview of Data Processing Workflow

- This workflow can be represented as a directed acyclic graphs (DAG). i.e. operators are represented as nodes and the data dependency between the operators are represented as edges in the graph.



The cylinder represent the input/output data and the circles represent the operators.

Figure 1: Example of Data Processing Workflow

Performance Metrics

- Depending on the requirements of different applications, there are different performance metrics in optimizing the MapReduce framework such as:
  - completion time,
  - monetary costs,
  - throughputs,
  - responsibility,
  - accuracy,
  - reliability, fairness etc.

tradeoff

- i.e. interactive applications → quick response and can sacrifice accuracy to some extend.
- batch applications are not very time sensitive and would like to minimize the monetary cost as long as it can finish within certain time period.

Performance Metrics

- **Completion time:** Completion time is the time elapses from the start of the workflows to the time when all the results are generated. It is the most important cost metric that represents the total time it takes to complete a workflow.
- **Monetary cost:** The monetary cost to complete the entire workflow based on cloud platforms (e.g, Amazon EC2) is also an important metric as the resources are claimed on demand and will be charged as long as you are using it. The monetary cost is closely related to the completion time. However, they not always correlated due to the pricing scheme in the cloud.
- **Response time:** Response time here represents the time elapses from the start of the workflows to the time when the system response. It is different from completion time as the response may contain only partial results. Short response time is especially important for interactive applications since the clients prefer short response time and are usually tolerant with incomplete results.

Factors Affecting Performance Metrics

- **Initial data scan costs**. It represents the time spent in loading data before the processing. Such initial data scanning requires reading and transferring data from the underlying distributed file system to the local place of the processing instance and recent studies have shown that initial scanning could take large fraction of the total completion time.
- **Intermediate data costs**. Intermediate data represents the data generated from the previous operator and will be consumed by the next operator in the workflow. For ease of fault tolerance, most parallel dataflow programming framework materialize these intermediate data when they are generated and delete them after the completion of the workflow. The I/O costs to materialize such data could significantly affect the completion time.
- **Data transmission costs.** The transmission of intermediate data between operators consists a considerable amount in the total I/O costs. Different data transmission model (e.g. pipelined and blocked) could also affect the completion time.
- **Scheduling affect.** The parallel execution of a data processing workflow needs to schedule the operator to different cluster nodes. (i.e, mapping between operators and the instances). Different scheduling strategies here can be applied which will lead to different completion time or monetary costs.

MapReduce: Workflow Processing Framework

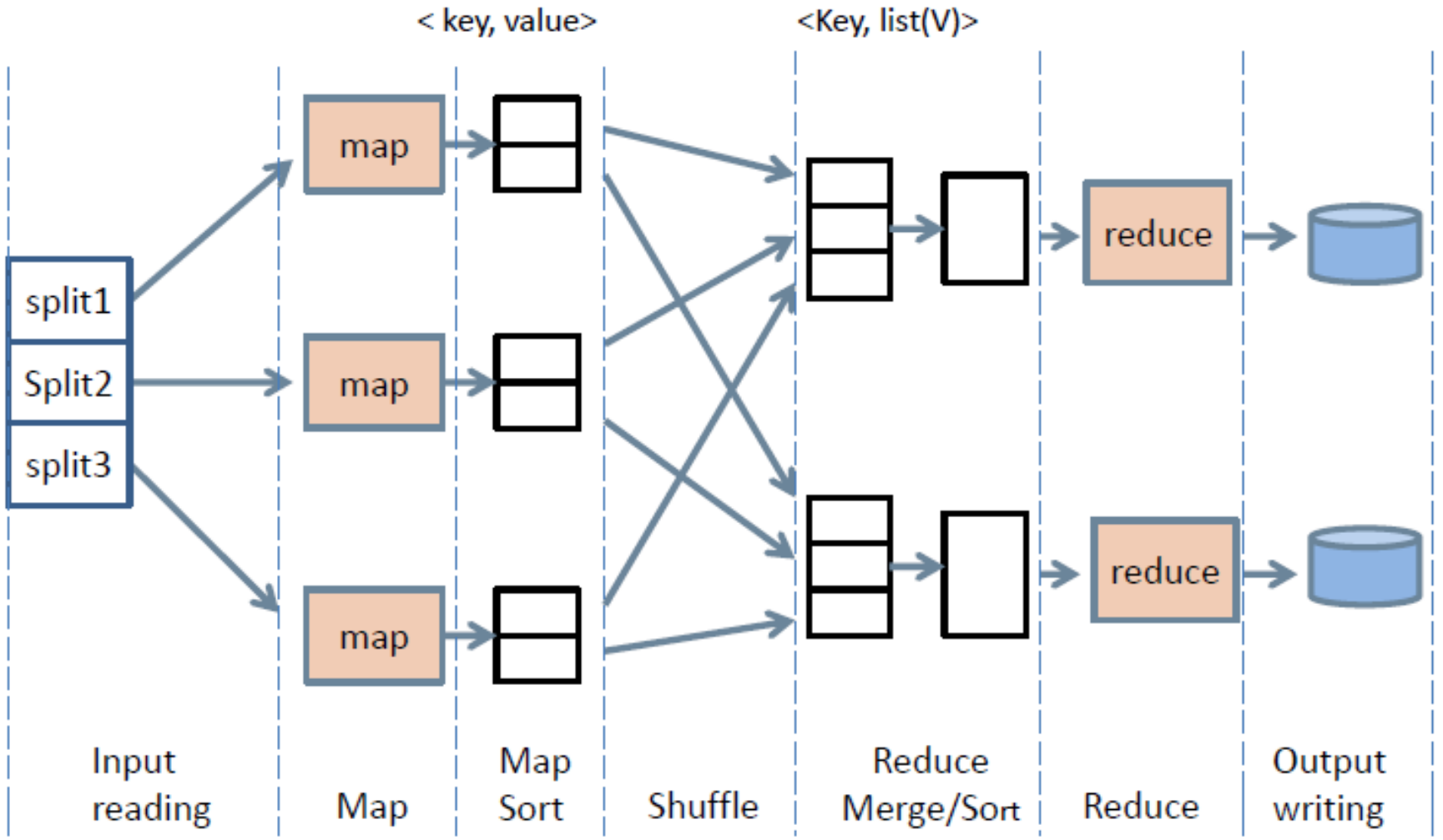
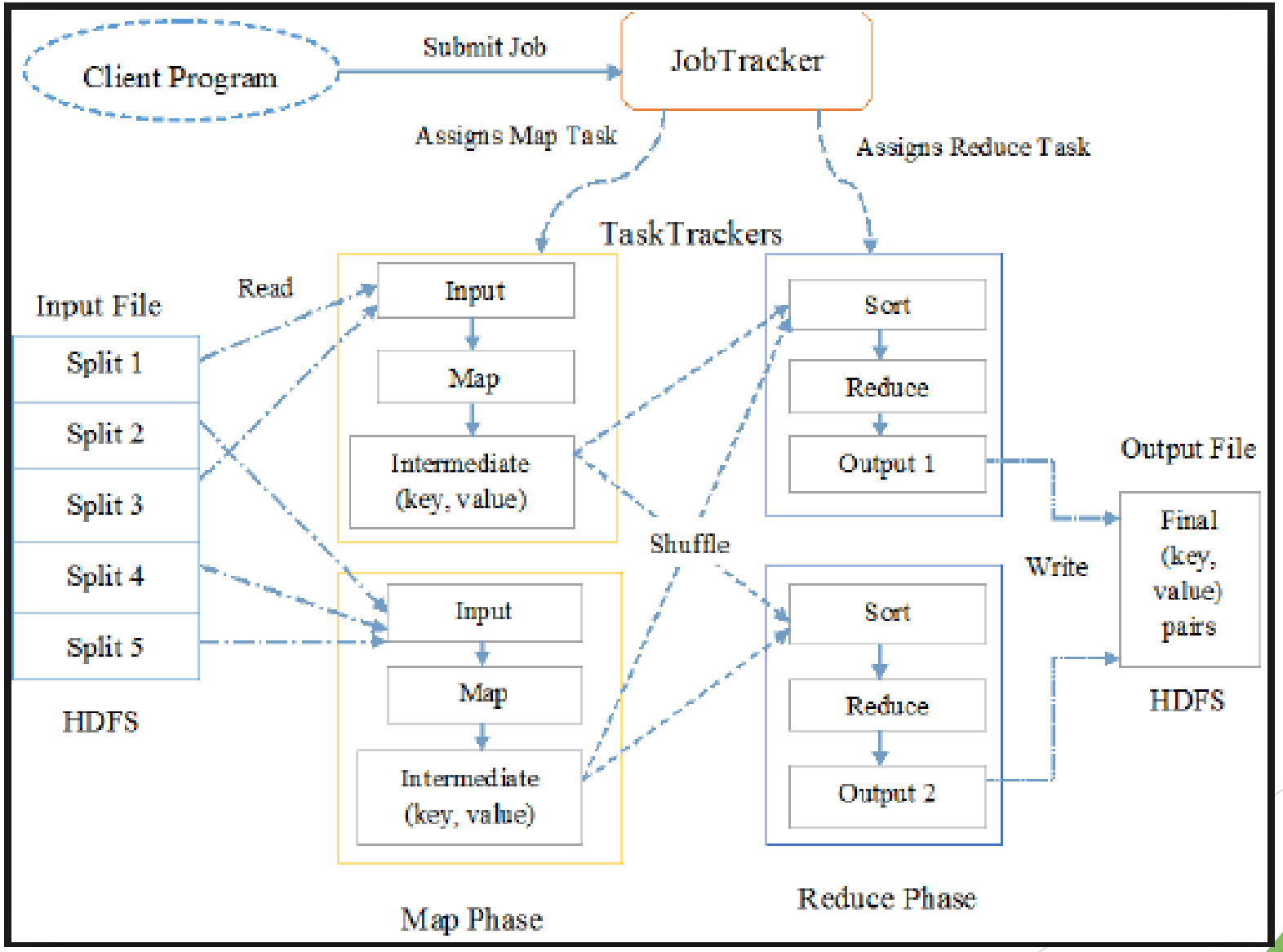


Figure 2: Architecture of MapReduce framework

MapReduce: Workflow Processing Framework



MapReduce: Workflow Processing Framework

- The execution of a MapReduce job is controlled by the master node which is called the JobTracker.
- There is one centralized JobTracker for the entire cluster and the rest instances are workers.
- The workers periodically connect to the JobTracker to report its current status and the available resources.
- The JobTracker is responsible for using the reported information for scheduling task to workers with available resources, monitoring the status of the task execution and handling failures and speculative executions.

Batch Oriented Data Analytics Processing

- Yahoo!, contributed to MapReduce which aims at batch data processing.

"Pull" based data transfer model The Hadoop MapReduce implementation adopts a "Pull" based data transfer model with more details as follows.

- In the map stage, each map task applies the map function to generate intermediate key/value pairs.
- It sorts and buffers these intermediate data and spill them into the local disk when the buffer is full.
- After all the input data are processed, the map tasks perform a merge sort of all the spilled files to generate a single final file.

Batch Oriented Data Analytics Processing

- In the reduce stage, the reduce tasks will fetch the intermediate data by copying them from the corresponding local files from all the completed map tasks.
- After all the data are fetched, it merges and sorts them into a single file and applies the reduce function to the merged data to generate the final output.

Batch Oriented Data Analytics Processing

Pull based Fault Tolerance

Since the MapReduce framework is typically deployed in the cloud environment (Cluster) which is based on large number of commodity machines, failures become quite common in that case. As a result, fault tolerance becomes an important function of the MapReduce framework. With the pull based data communication model, the Hadoop MapReduce provides simple and automatic failure recovery strategy.

- **Task failure** The JobTracker receives periodical report from each worker which contains the statue for all the tasks assigned to that worker. Once a task failure is reported, the JobTracker simply re-executes the failed task on a different worker. For map tasks, when the failed task re-executes and generates new map outputs, the reduce tasks will be informed to fetch the new data from that task. For reduce task, as all map outputs are materialized in the local disk, the reduce task can fetch again the corresponding data from all the completed map tasks and re-execute the reduce function on them.

Batch Oriented Data Analytics Processing

Pull based Fault Tolerance

- Worker failure As the JobTracker receives periodical report from each worker, if no response is received from a worker in a certain amount of time, the JobTracker marks the worker as failed. All the map tasks assigned to that worker (including those which are already completed) need to be re-executed since the map outputs are stored in the local machine and is lost when the worker failed. The JobTracker also re-executes all the incomplete reduce task on the failed worker.
- **Master failure** According to the design of the MapReduce framework, there is only a single JobTracker, its failure is unlikely and no particular recovery strategy is proposed in the Hadoop implementation. However, it is easy to make the JobTracker write periodic checkpoints of the master data structures. If the JobTracker dies, a new copy can be started from the last checkpointed state.

Continuous Data Analytics Processing

Hadoop Online Prototype (HOP): an alternative implementation of the MapReduce framework that provides directions to support interactive and continuous applications.

"Push" based data ow model

- Instead of the Hadoop MapReduce which materializes all the map outputs to the local disk before the start of reduce phase, HOP tries to "push" data directly from map to reduce tasks. Specifically, in the aggressive push model, when a client submits a new job, the JobTracker assigns the map and reduce tasks associated with the job to the available TaskTracker slots. To simplify the discussion, we assume that there are enough free slots to assign all the reduce tasks for each job. Each reduce task contacts every map task upon initiation of the job, and opens a TCP socket which will be used to pipeline the output of the map function. When an intermediate key/value pair is produced, the map task determines which partition (reduce task) the pair belongs, and immediately sends it via the appropriate socket.
- The problem with such data transmission model is that first it can not make use of the combiner function which could significantly increase the amount of data transferring between map and reduce stage. A related problem is that as there is no sort work in the map side, it pushes more work on the reduce side which may lead to longer reduce phase. It will also bring problems for fault tolerance since no intermediate data is materialized, whenever a reduce task fails.

Continuous Data Analytics Processing

- To address these problems, HOP proposes a refined data transmission strategy. Instead of sending the output to reducers directly, it buffers the map outputs until it grows to a threshold size. The map task then applies the combiner function, sorts the output and spills the buffer contents to disk. If the reduce tasks have enough resources to process the outputs, the spill file will then be sent to the reduce tasks. However, if the reduce tasks fall to keep up with the production of the map tasks, the map task accumulates multiple spill files until the reduce tasks catch up the speed. Then it merges the accumulated spill files into a single file, applies the combiner function on it and then and sent the combined results to the reducers.
- HOP also supports data pipelining between jobs with a workflow. Similar to the pipelining between the map and reduce stage, the reduce tasks of the previous job can pipeline their output directly to the map tasks of the next job, sidestepping the need for waiting for the completion of the reduce stage of previous jobs

Continuous Data Analytics Processing

Push based Fault tolerance

In HOP, the map outputs are sent to the reduce tasks in a pipelined fashion, such implementation makes it more difficult to recover from failures and new techniques need to be introduced to achieve fault tolerance.

- **Map task failure** To recover from map task failures, the reduce tasks keep the record which map task produces each pipelined spill file. The reducer treats the output of a pipelined map task as "tentative" until the map task has committed successfully. The reducer only merge together spill les generated by the same uncommitted mapper. Thus, if a map task fails and re-executes, each reduce task simply ignore any tentative spill les produced by the failed map task and starts to receive data generated by the re-executed map task.
- Reduce task failure If a reduce task fails and a new copy of the task is started, the new reduce instance must collect all the data that was sent to the failed reduce attempt. To enable reduce failure recovery, map tasks retain their output data on the local disk for the complete job duration. This allows the map outputs to be resent if any reduce tasks fail. Note given such implementation, the I/O costs of HOP is the same as the Hadoop MapReduce. However, the key advantage of HOP is that the start of reduce tasks is not blocked waiting for the completion of the map tasks.