Big Data Technologies

Agenda

- Apache Hadoop
  - MapReduce
  - YARN

Hadoop

Map-Reduce Programming

MR Job XML Configuration

- fs.defaultFS = hdfs://namenode:9000/
- mapreduce.framework.name = yarn
- yarn.resourcemanager.address = resourcemanager:8032
- Other config parameters can be set up as per application requirement e.g. dfs.replication, dfs.blocksize, or additional user defined parameters.
- XML configuration is given using "-conf" option.

Understanding MR execution

- MR summary (on console)
  - Number of mapper & reducer tasks.
  - Number of input & output records for mapper
  - Number of input & output records for reducer
- MR log files review
  - $HADOOP_HOME/logs
  - MR jobs logs --> $HADOOP_HOME/logs/userlogs
  - The MR job logs --> $HADOOP_HOME/logs/userlogs/application_appid
  - The MR mapper/reducer logs --> $HADOOP_HOME/logs/userlogs/application_appid/container_id

- stdout/stderr
- syslog

Custom Job counters

- Hadoop Counters provides a way to measure the progress or the number of operations that occur within map/reduce job.
- Counters in Hadoop MapReduce are a useful way for gathering statistics about the MapReduce job for quality control or for application-level.
- Example:
  - step 1: Create enum of custom Job counter

```
enum NcdcCounters {
    BAD_RECORDS, BAD_READINGS, GOOD_READINGS
}
```

  - step 2: Access the counter in Mapper or Reducer program and increment it.

```
Counter cntr = context.getCounter(NcdcCounters.GOOD_READINGS);
cntr.increment(1);
```

Input/Output Format and Input Splits

- InputFormat – how to read data
  - FileInputFormat, TextInputFormat
  - KeyValueTextInputFormat, NLineInputFormat
  - DBInputFormat
  - CombineTextInputFormat
- RecordReader (nested class in InputFormat) – Logical division of record

- Number of mappers = Number of input splits
- Number of input splits ≈ Number of HDFS blocks
- Typically mapper tasks are executed on the system on which data blocks (or replica) are present. Such mapper tasks are called as "Data-local Map tasks".
- OutputFormat – how to write data
    - FileOutputFormat, TextOutputFormat
    - DBOutputFormat
- RecordWriter (nested class in OutputFormat) – Write individual record
- Number of output files = Number of reducers
- Output written on HDFS (replicated)

Data flow of MR job (Single reducer)

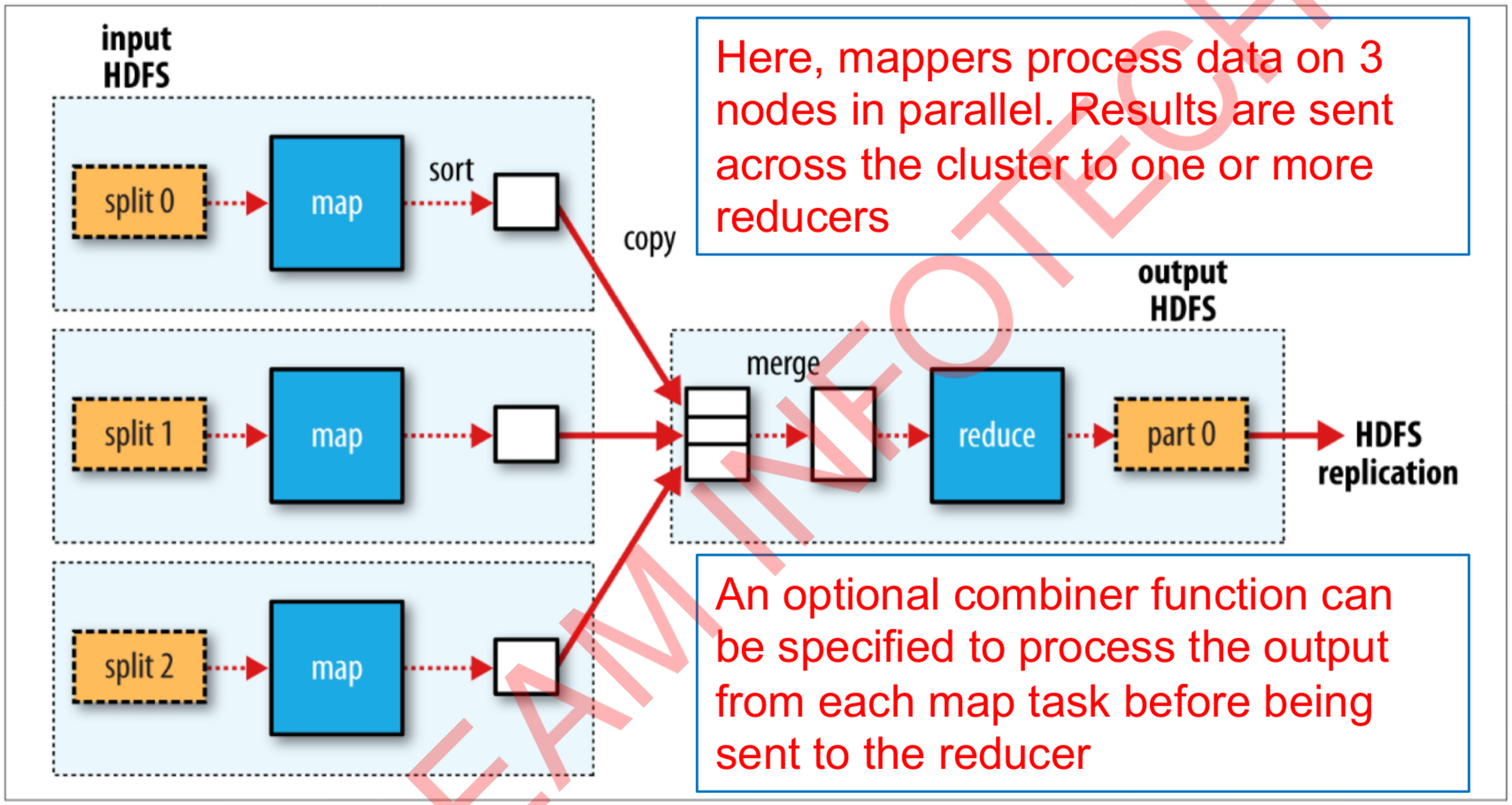
- 
input HDFS
split 0
map
sort
split 1
map
split 2
map
copy
merge
reduce
part 0
output HDFS
HDFS replication
Here, mappers process data on 3 nodes in parallel. Results are sent across the cluster to one or more reducers
An optional combiner function can be specified to process the output from each map task before being sent to the reducer

Figure 2-3. MapReduce data flow with a single reduce task

Combiner

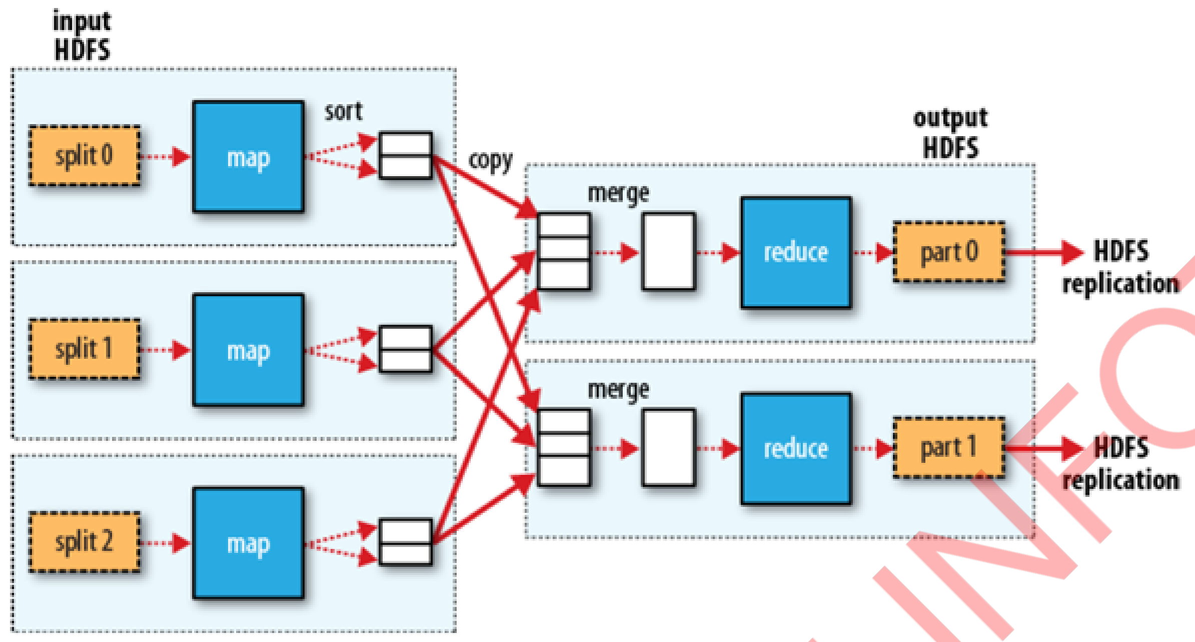
- Combiner is a local reducer i.e. runs reducer (aggregation logic) within mapper task process.
  - Minimize output of mapper task
  - Less merge & shuffle
  - Less network transfer

- Less aggregation in reducer
- Combiner is optional.
- Works for commutative & associative aggregate functions only.
  - A + B = B + A
  - A + (B + C) = (A + B) + C

Partitioner

- By default MR job have single reducer.
- Having huge data for aggregation may lead to out of memory error.
- Number of reducers can be configured in job configuration file OR in driver code.
  - job.setNumReduceTasks(2);
  - mapreduce.job.reduces = 2
- Number of partitions = Number of reducers
- Output of mapper is divided into multiple partitions based produced key.
- By default HashPartitioner is used, that distributes mapper output in number of partitions uniformly. Refer slides.
- Can implement custom partitioner class (inherited from Partitioner class), if HashPartitioner is not suitable (not balance load on reducers).
  - job.setPartitionerClass(CustomPartitioner.class);

Data flow of MR job (Multiple reducer)

- 

Hadoop MR data flow (detailed)

- Mapper Task/Process
  - HDFS --> InputFormat+RecordReader --> Mapper --> Partitioner --> Ring Buffer (100MB) --> Sort & Group --> Combiner --> Write on Disk.
- Reducer Task/Process
  - Mapper Task Output --> Reducer input buffer --> Merge Sort & Group --> Reducer --> OutputFormat+RecordWriter --> HDFS.
- Refer slides
- https://0x0fff.com/hadoop-mapreduce-comprehensive-description/

Map-only Jobs

- If data processing includes only data cleaning/filtering (per record) and no aggregation operations, we create Map-only job.
- Map only job is created by setting number of reducers to zero.
  - job.setNumReduceTasks(0); // Java code -- OR
  - mapreduce.job.reduces = 0 // Java xml config file

- Applications
  - Data ingestion (from different sources)
  - Data cleansing
  - Data filtering
  - Data pre-processing
- SQOOP is Hadoop eco-system/application that generates Map-only jobs for transferring data from RDBMS to Hadoop and vice-versa.
  - RDBMS --> sqoop import --> Hadoop
  - Hadoop --> sqoop export --> RDBMS

Reduce-only Jobs

- It is not possible to have Reduce only jobs.
- However, we can implement Map-Reduce jobs where mapper is not doing any significant processing. The output of mapper will be same as input data. Such mapper is called as IdentityMapper.
- The output of IdentityMapper will be sorted (by null key) and sent to the reducer. The reducer code should be written accordingly.
- This is not commonly used.