What is common between all of them?

Data-intensive computing with MapReduce!
Map Reduce - Introduction

- Framework for parallel computing
- Prominent parallel data processing tool
- Uses clustered resource
- Very good level of abstraction for the programmers, and do not have to deal with issues of parallelization, load balancing, fault tolerance

Data Analysis

- Map and Reduce functions
- Map: gives a key value pair
- Reduce: for each unique key
Step 1: Splitting the Input

Large data is initially divided into large number of smaller portions. The data is divided such that we have splits equal to the number of worker machines, thus each worker has something to work on.

Step 2: Master and worker co-ordination.
Step 3: Mapping by each worker

Each worker now starts generating key value pairs of the data assigned to them. So the map function gets rid of all the irrelevant data and just passes on the key value pairs of the data intended to be filtered or sorted, thus linearly scaling the performance because of the parallelism.

Step 4: Partitioning within the workers
The partitioning is responsible for segregating the data further. The function is simply hash of key modulo. It becomes easier in the reduce stage.

Step 5: Reduce Sort

The map workers are done with their work and now the reduce workers are notified to start working using the data returned by the mapping worker. The reduce worker contacts every map worker via remote procedure calls to get the \((key, value)\) data that was targeted for its partition. This data is then sorted by the keys. Thus at the end of this step we have the data with the same key grouped together.
Step 6: Final reducing step

Thus this step returns the required goal of running the map reduce file. This can be finding a word from large file data or studying the web search logs, or any kind of data processing.

These functions are run on Distributed file systems, like GFS, AFS, etc.
Understanding Map Reduce better
Lisp Map & Reduce

- The map function takes a function and a set of values as a parameter.
- This function is then applied to each of the value from the data.
  \[(\text{map 'length '((()) (a) (ab) (abc)))}\]
- The above function applies this function to each of the values. Length of the values are returned.
- Now the ‘reduce’ function is given a binary function and a set of values. All the returned values are thus combined.
Google’s Observation

- Key Word Search

MAP: The basic working of the search engine that is finding the key word and returning the URL.

REDUCE: This stage combines all the resulting URLs which have the keyword and return it.
Google’s need for MapReduce

- Client request of logs and web resources, largest Search Engine
- Large derived data
- Easy calculation, large processing
- Processing distributed amongst machines
- Google’s new abstraction allowing simple computations, hiding messy parallelism details
Hadoop- Open Source for Map Reduce

- Google’s patent
- Not an open source
- Hadoop creation
- Basis of Hadoop and MapReduce
Figure 1: Hadoop Architecture

Scheduling in Hadoop

- Two Types: Fair Scheduling and Capacity Scheduling
- Fair Scheduling: works when we have Single queue of jobs
  - Equal share of physical resources
  - Single MR job, occupies the whole of the cluster
- Capacity Scheduling: More sophisticated type of scheduling
  - Multiple queues can work along
- Each process in each queue is guaranteed to get the cluster resource
- MRShare: Framework for sharing multi query executions in MapReduce
- finds an optimal way of grouping a set of queries using
dynamic programming.
- transforms a batch of queries into a new batch that will be executed more efficiently
- merging jobs into groups and evaluating each group as a single query.
- Suppose $|D|$ is the size of input data that $n$ MR jobs share
- Complexity of sorting the combined mapped output of all jobs will be $O(n \cdot |D|\log(n \cdot |D|))$
Some problems where MapReduce has been used

- Distributed grep (search for words)
  1. *Map*: emit a line if it matches a given pattern
  2. *Reduce*: just copy the intermediate data to the output

- Count URL access frequency
  1. *Map*: process logs of web page access; output
  2. *Reduce*: add all values for the same URL
Debates

- Major step backwards in parallel processing compared to DBMS
- Hadoop scalable but achieves very low efficiency
- Hadoop stands out in the “Gray Sort Benchmark test” for 100TB sorting.
- Not a cheap solution
- Cost of maintaining cluster difficult
- Increases fault tolerance
Advantages

- Simple and easy to use
- Flexible
- Independent of the storage
- Fault tolerant
- High scalability
Limitations of MapReduce

- Low efficiency
- Cannot be used when computations depend on previously calculated values
- Can handle large data sets but constraints program’s ability smaller data items
Software in place of Hadoop and MapReduce

- DISCO
- Greenplum
- Aster Data
Companies who have adopted similar algorithms

- A9.com
- AOL
- Facebook
- The New York Times
- Last.fm
- Baidu.com
- Joost
- Veoh
References

- http://www.dbms2.com/2008/01/18/the-great-mapreduce-debate/
- https://www.cs.rutgers.edu/~pxk/417/notes/content/mapreduce.html