阅读 MapReduce 1-5页

Motivation

MapReduce is a programming paradigm created to save programmer from messy details of parallelization fault-tolerance data, data distribution and loadbalancing.

It is inspired by map and reduice function in LISP language.

The writer find that their work can be abstracted as

- **MAP**: Iterate over all records to get immediate (key, value) pairs.
- **Reduce**: Merge the immediate values with the same keys to get a smaller set of values.

Values are accessed by iterator to avoid lists with too large value.

Examples

word counting Example:

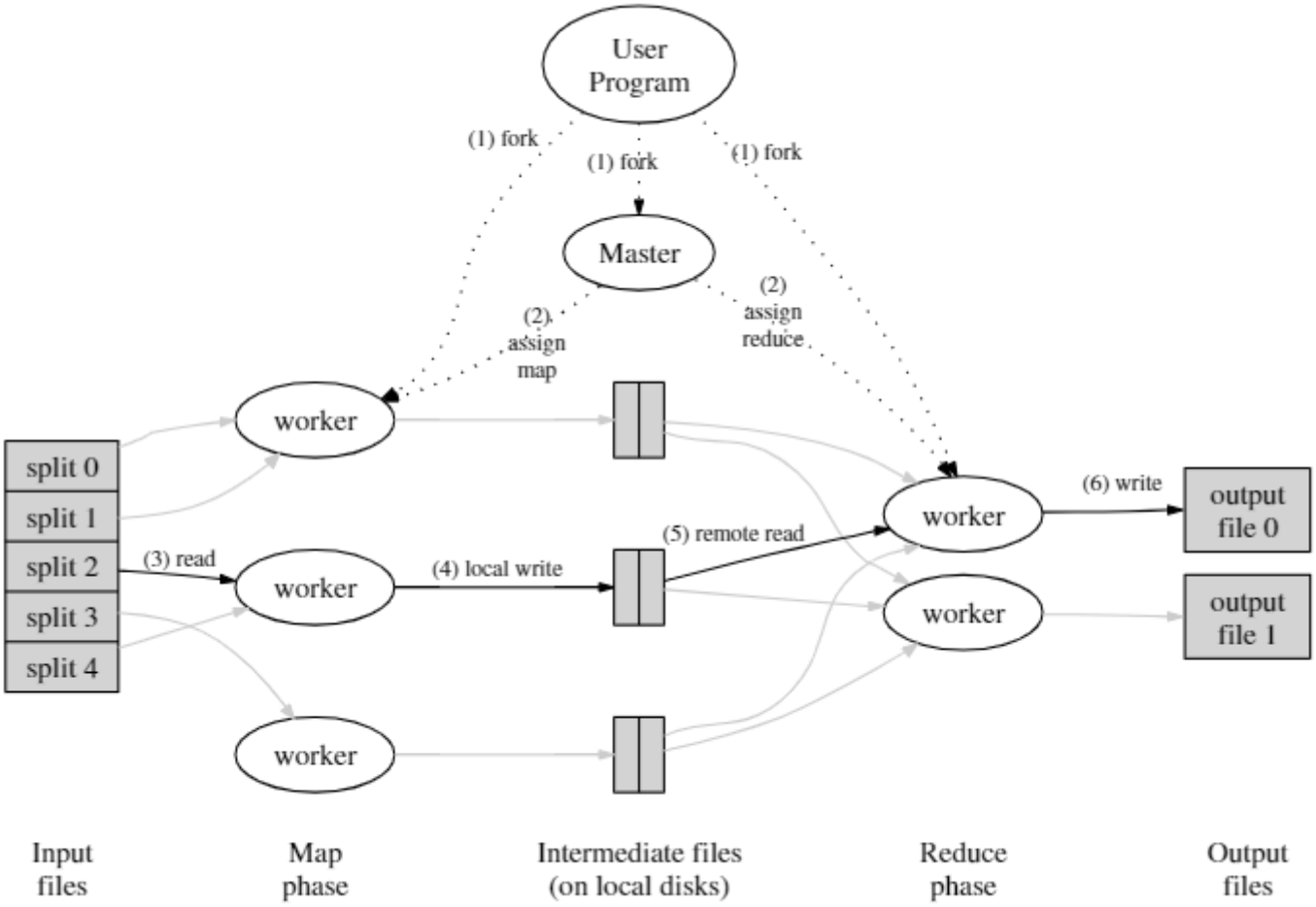
```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Other examples of programs in MapReduce paradigm can be find in 2.3:

1. Distributed Grep
2. Count of URL aceeess Frequency
3. TO BE FILLED

Implementation Details:

Execution Overview



1. **Split**: Split the input files into M pieces(hyberparameter typically 16MB or 64MB per piece).
2. **Assignment**: Master assign each idle machine a map or reduce task.
3. **Map**: Writing buffered pair to **local** disk.
4. **Reduce**:
    - Get remote buffered data from map workers.
    - Sort all pairs by keys
    - Reduce
5. **Completion**: Return to user code.

Fault Tolerance

Workers

- Complete map Fault worker will be reset.(Why no reduce?)
- Map or reduce task in progress will be reset.

Master

Aborts

Semantrics in the Presence of Failures

To be filled

Locality

Master attempt to assign map task on a machine that contains a replica of input data. Otherwise, it attempt to assign a nearest machine.