专题1： MapReduce的概念、原理与应用

谢磊 博士
南京大学计算机科学与技术系

主要内容：

一、MapReduce的应用背景

二、MapReduce的概念

三、MapReduce的原理

四、MapReduce的实现

五、MapReduce的性能

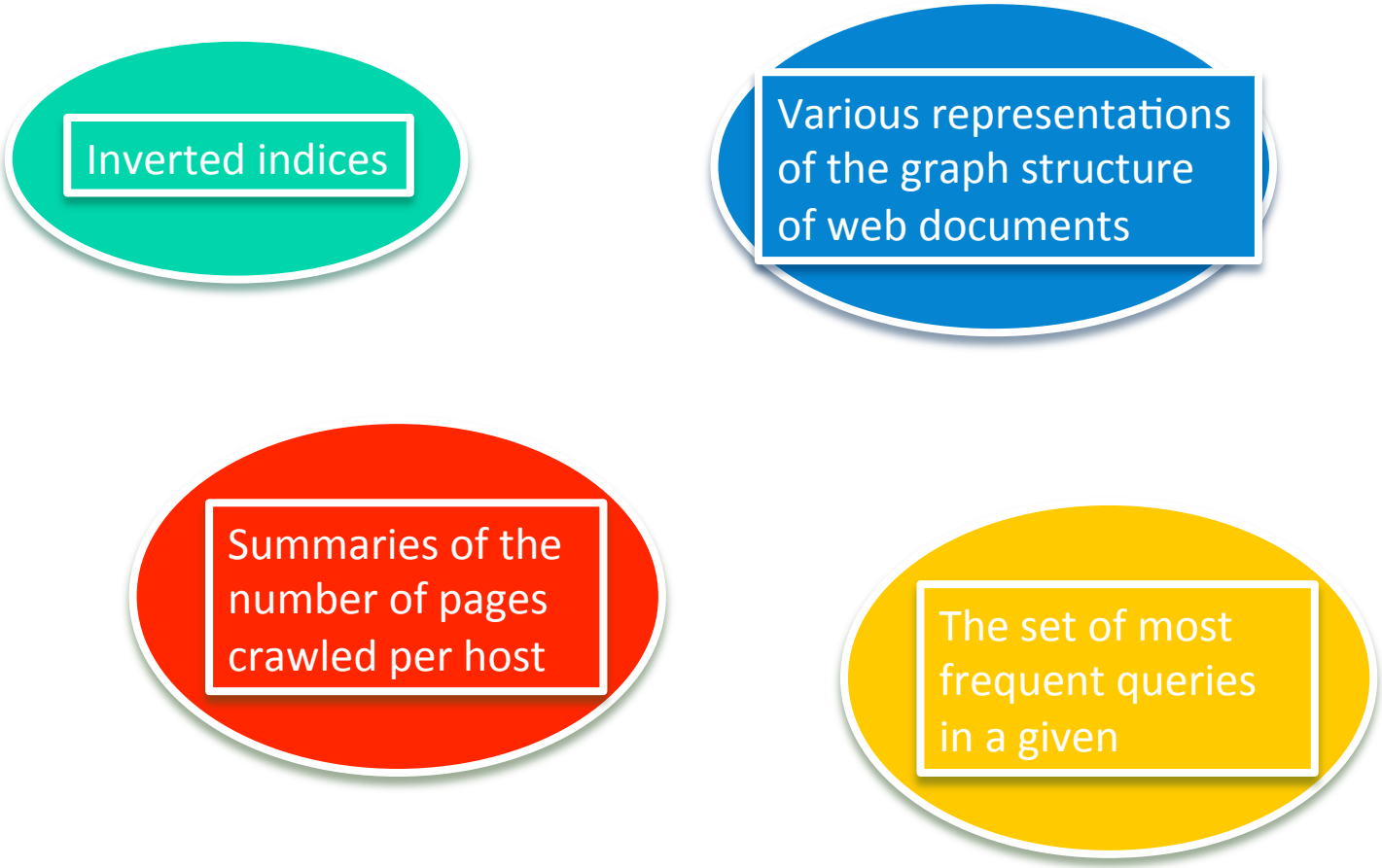
六、参考文献

MapReduce的应用背景-1

- Google have implemented hundreds of special-purpose computations that process large amounts of raw data,
  - such as crawled documents, web request logs, etc.,

MapReduce的应用背景-1

- Google's data center compute various kinds of derived data.



MapReduce的应用背景-2

- Most such computations are conceptually straightforward. However,
  - the input data is usually large
  - and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time.

数据总量	• 100~1000PB
数据处理量	• 10~100PB/天
网页	• 千亿~万亿
索引	• 百亿~千亿
更新量	• 十亿~百亿/天
请求	• 十亿~百亿/天
日志	• 100TB~1PB/天

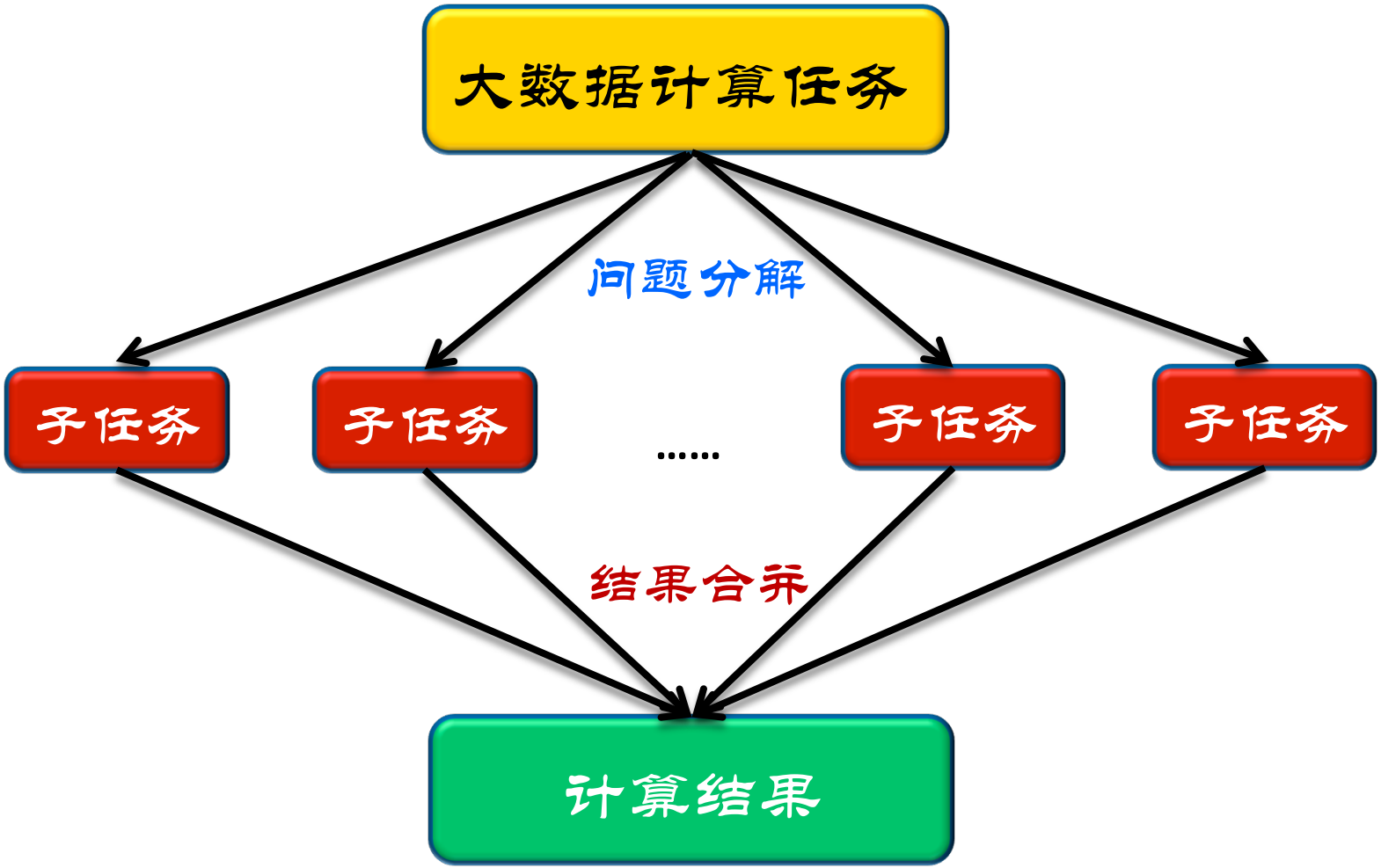
MapReduce的概念-1

- MapReduce
  - **MapReduce** is a software framework introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers.
  - The framework is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as their original forms.
  - MapReduce libraries have been written in C++, C#, Erlang, Java, OCaml, Perl, Python, PHP, Ruby, F#, R and other programming languages.

MapReduce的概念-2

- MapReduce is a framework for processing huge datasets on certain kinds of distributable problems using a large number of computers (nodes).
- The nodes collectively are referred to as
  - a cluster (if all nodes use the same hardware)
  - or a grid (if the nodes use different hardware).
- Computational processing can occur on data stored either in a filesystem (unstructured) or within a database (structured).

MapReduce的概念-2



MapReduce的概念-3

- **"Map" step**
  - The master node takes the input, partitions it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node.
- **"Reduce" step**
  - The master node then takes the answers to all the sub-problems and combines them in some way to get the output – the answer to the problem it was originally trying to solve.

MapReduce的原理-1

- Example

Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudo-code:

```
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));
```

MapReduce的原理-1

Then consider the problem of counting the number of occurrences of each word in the following sentences:

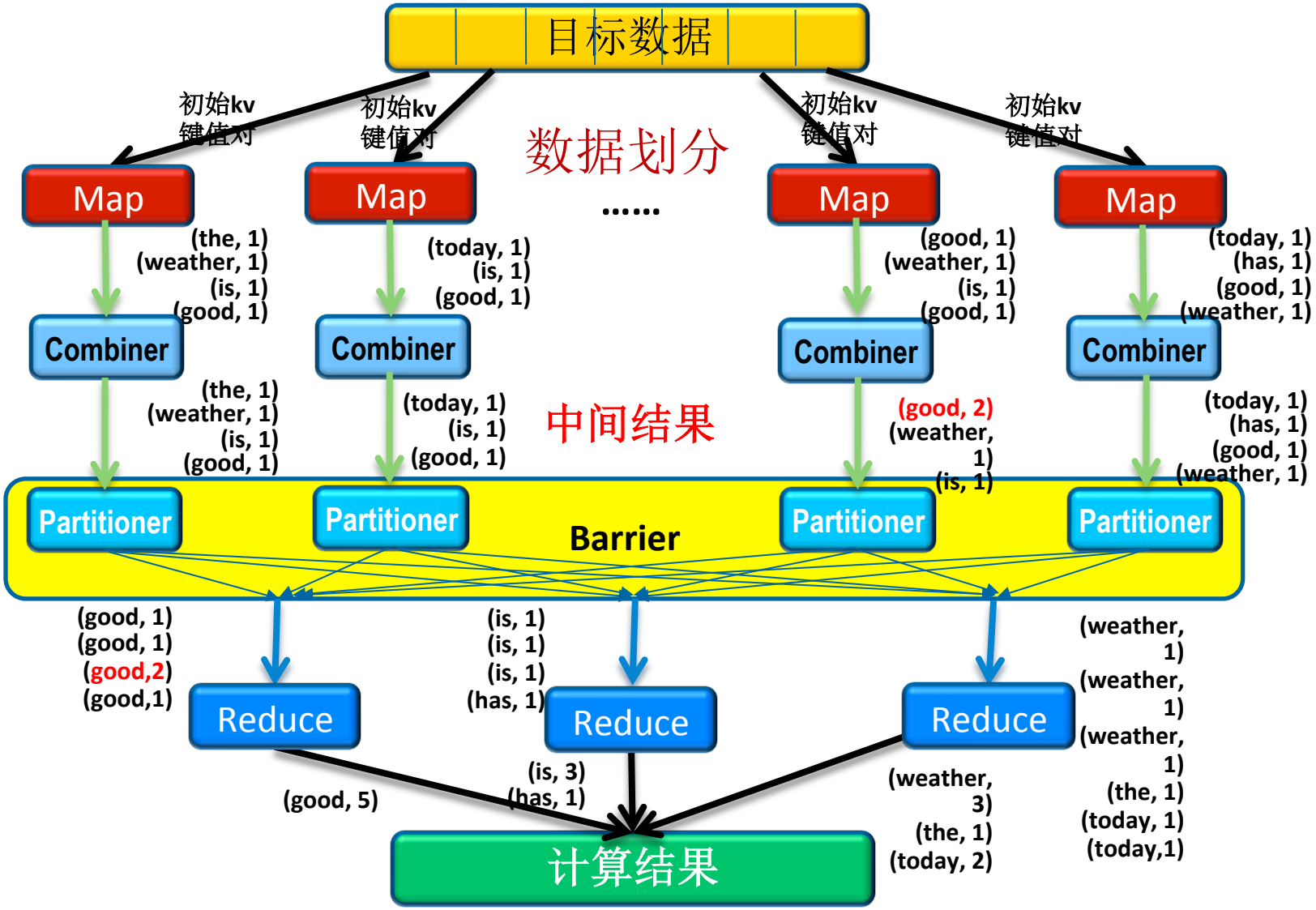
The weather is good.

Today is good.

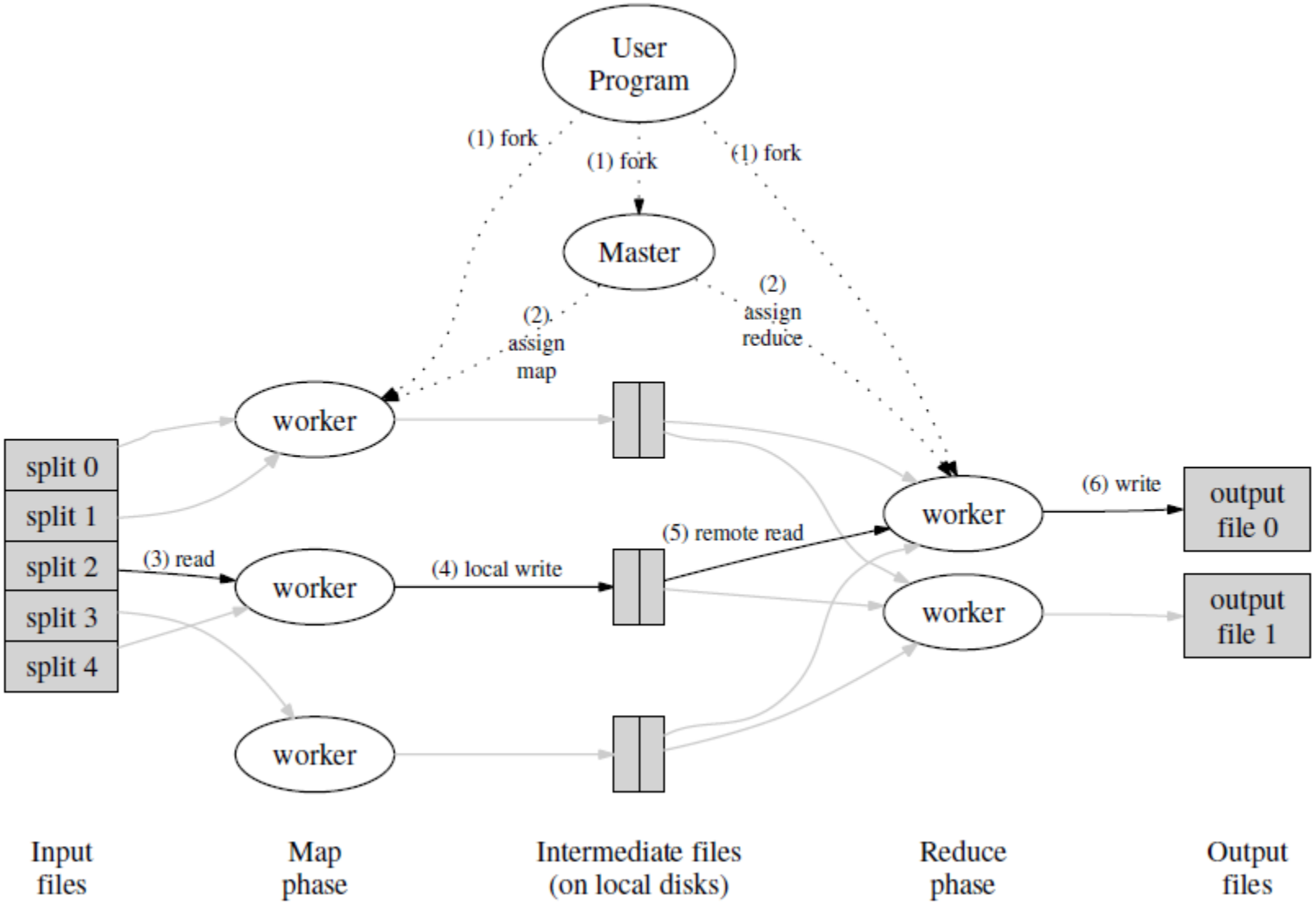
Good weather is good.

Today has good weather.

MapReduce的原理-1

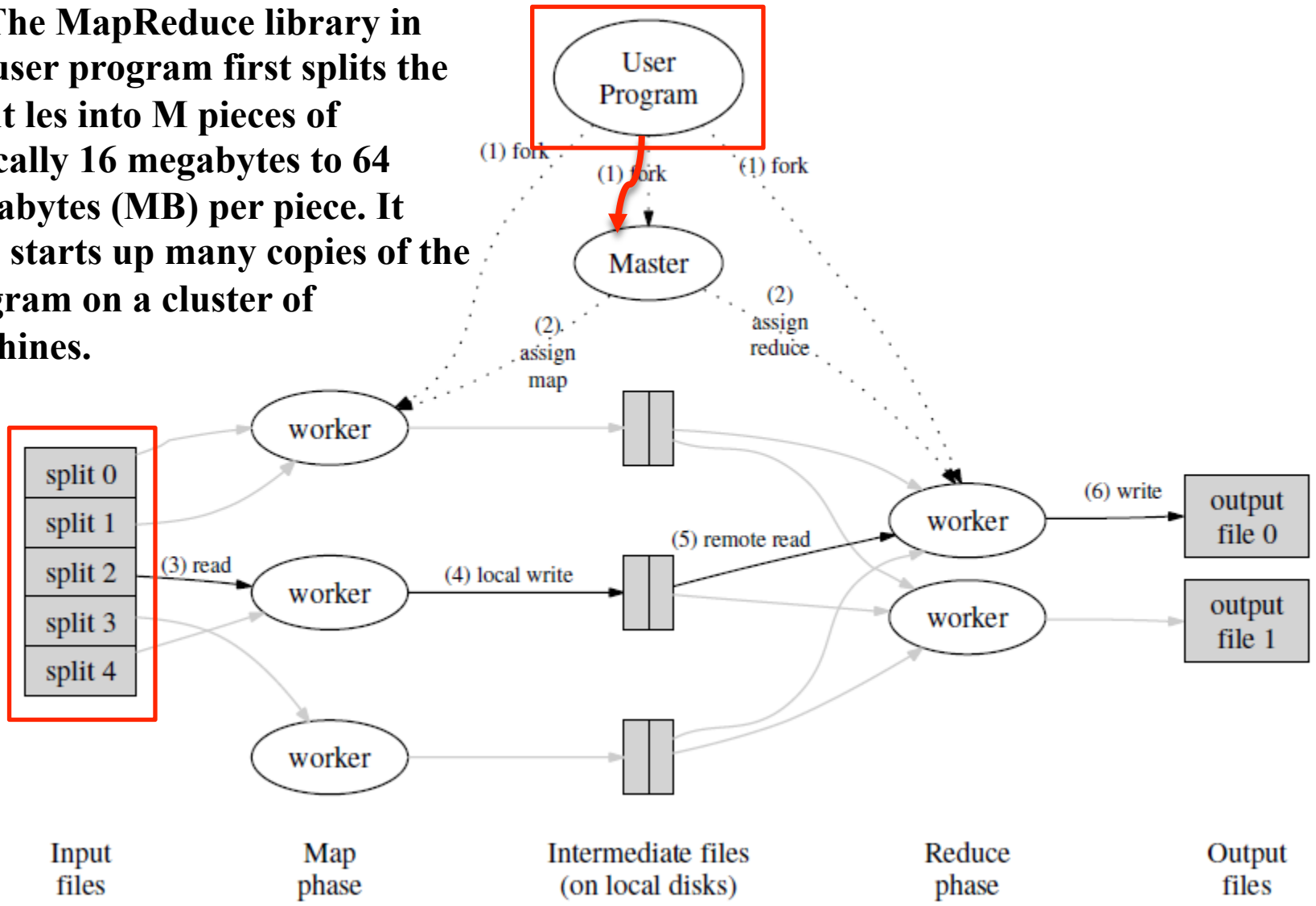


MapReduce的原理-2

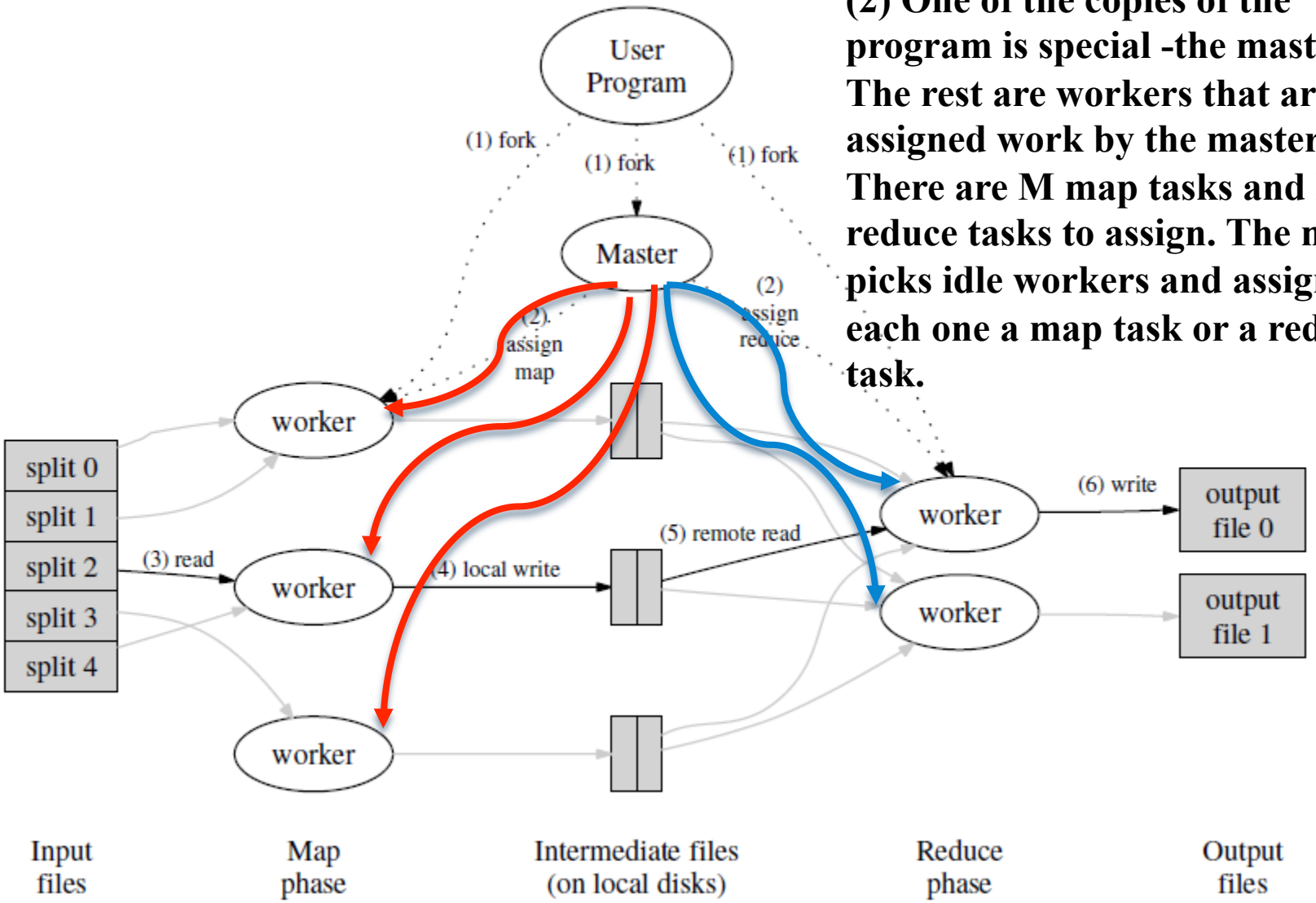


MapReduce的原理-3

(1) The MapReduce library in the user program first splits the input les into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece. It then starts up many copies of the program on a cluster of machines.



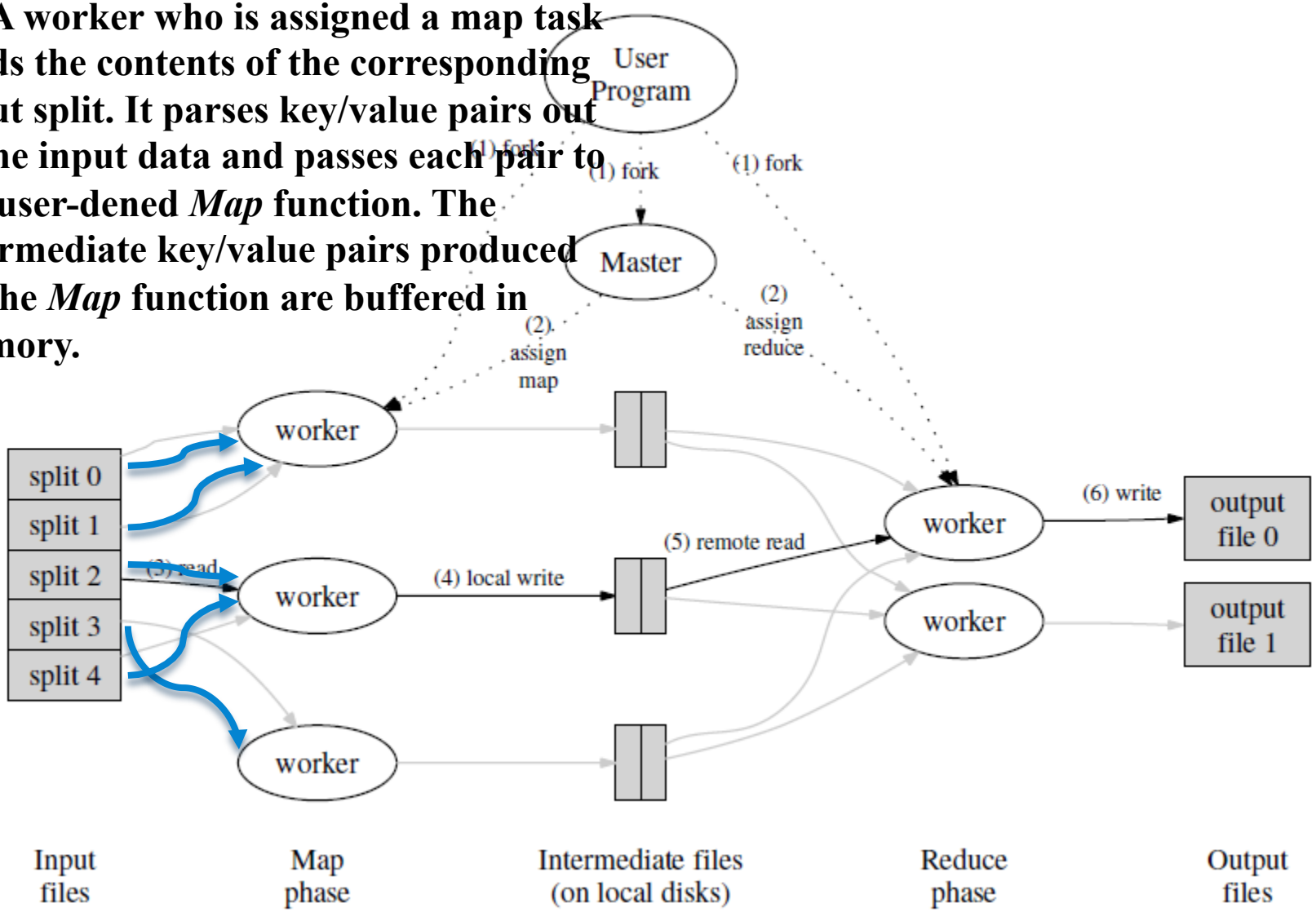
MapReduce的原理-4



(2) One of the copies of the program is special -the master. The rest are workers that are assigned work by the master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.

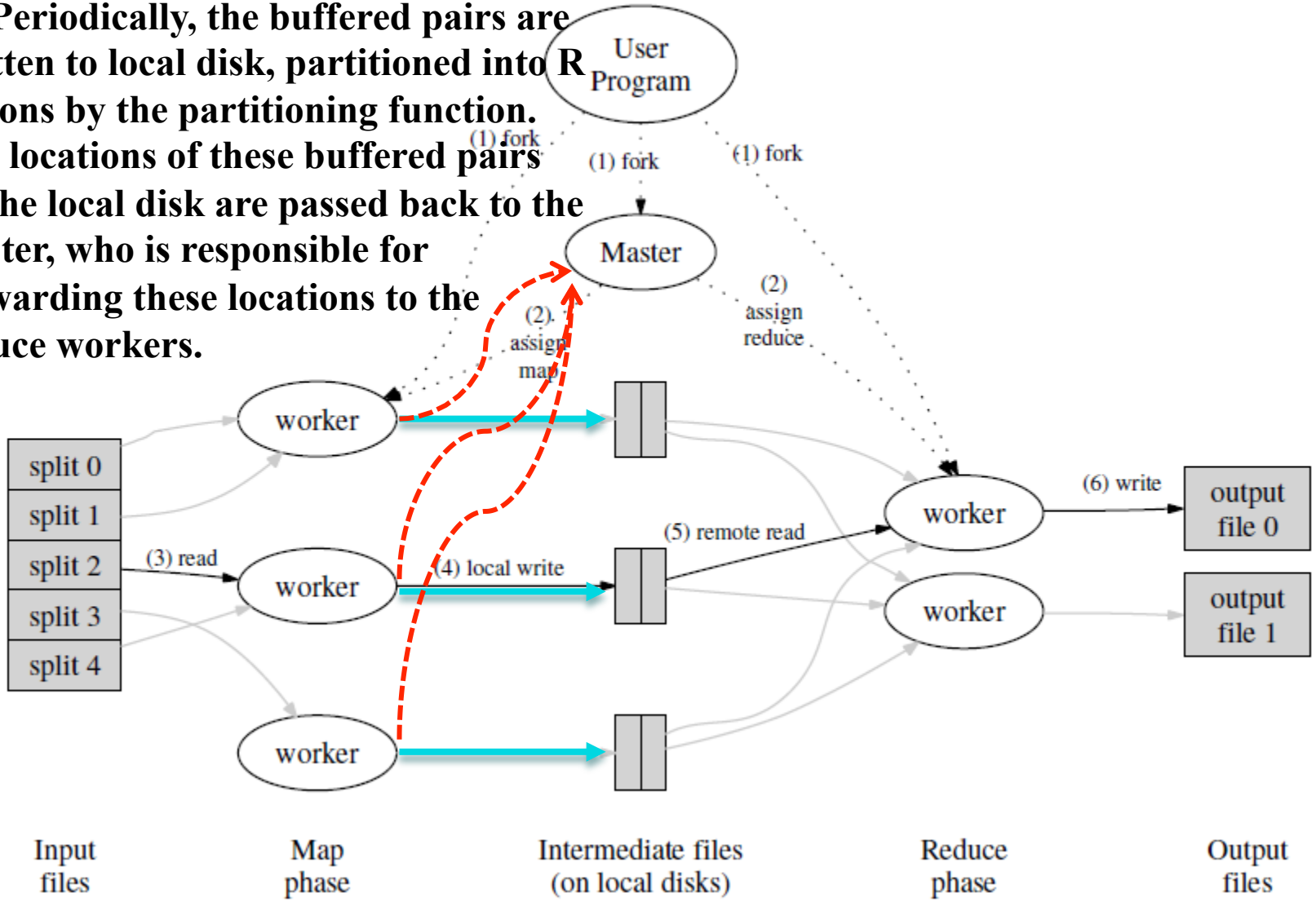
MapReduce的原理-5

(3) A worker who is assigned a map task reads the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-dened *Map* function. The intermediate key/value pairs produced by the *Map* function are buffered in memory.



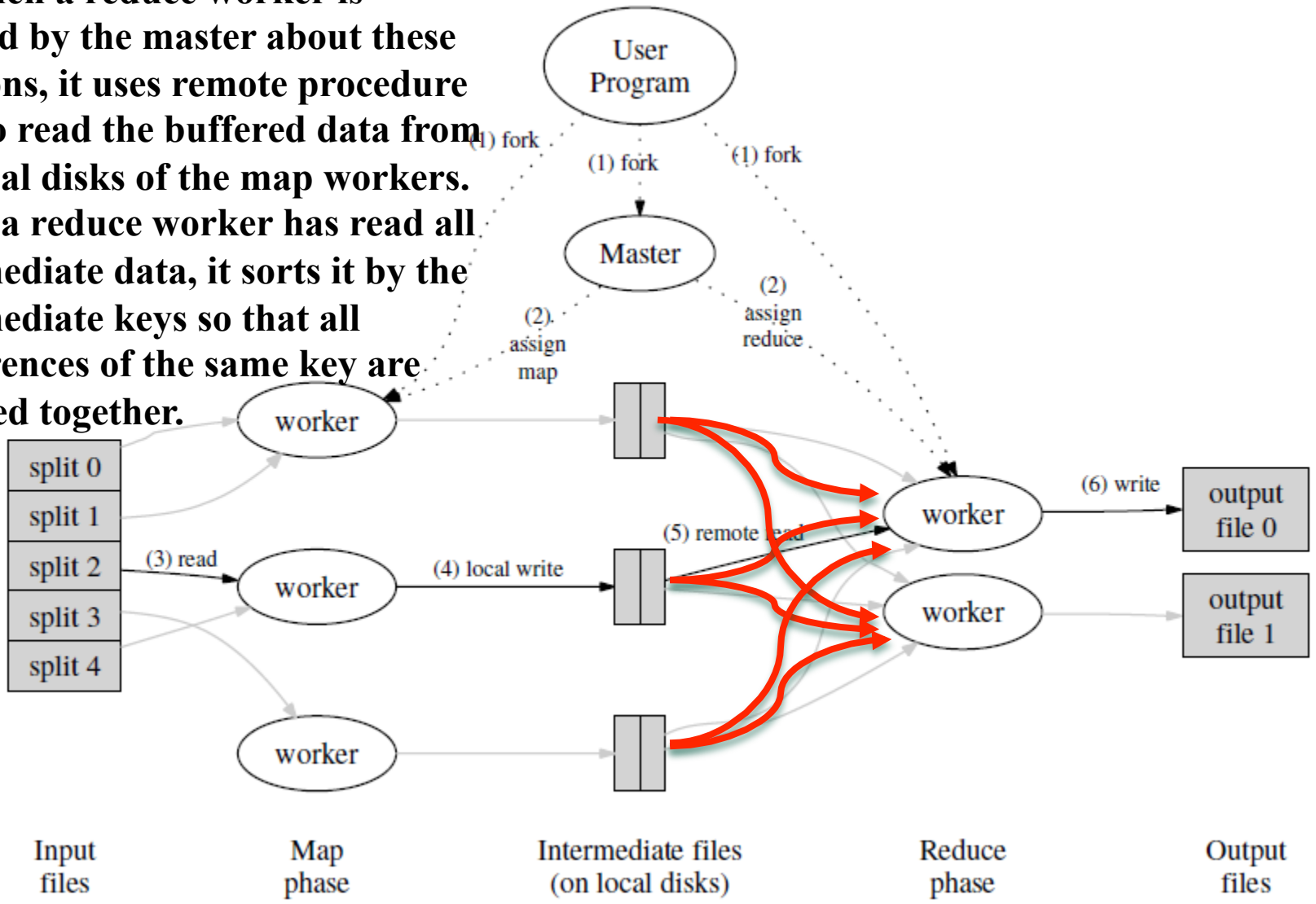
MapReduce的原理-6

(4) Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers.

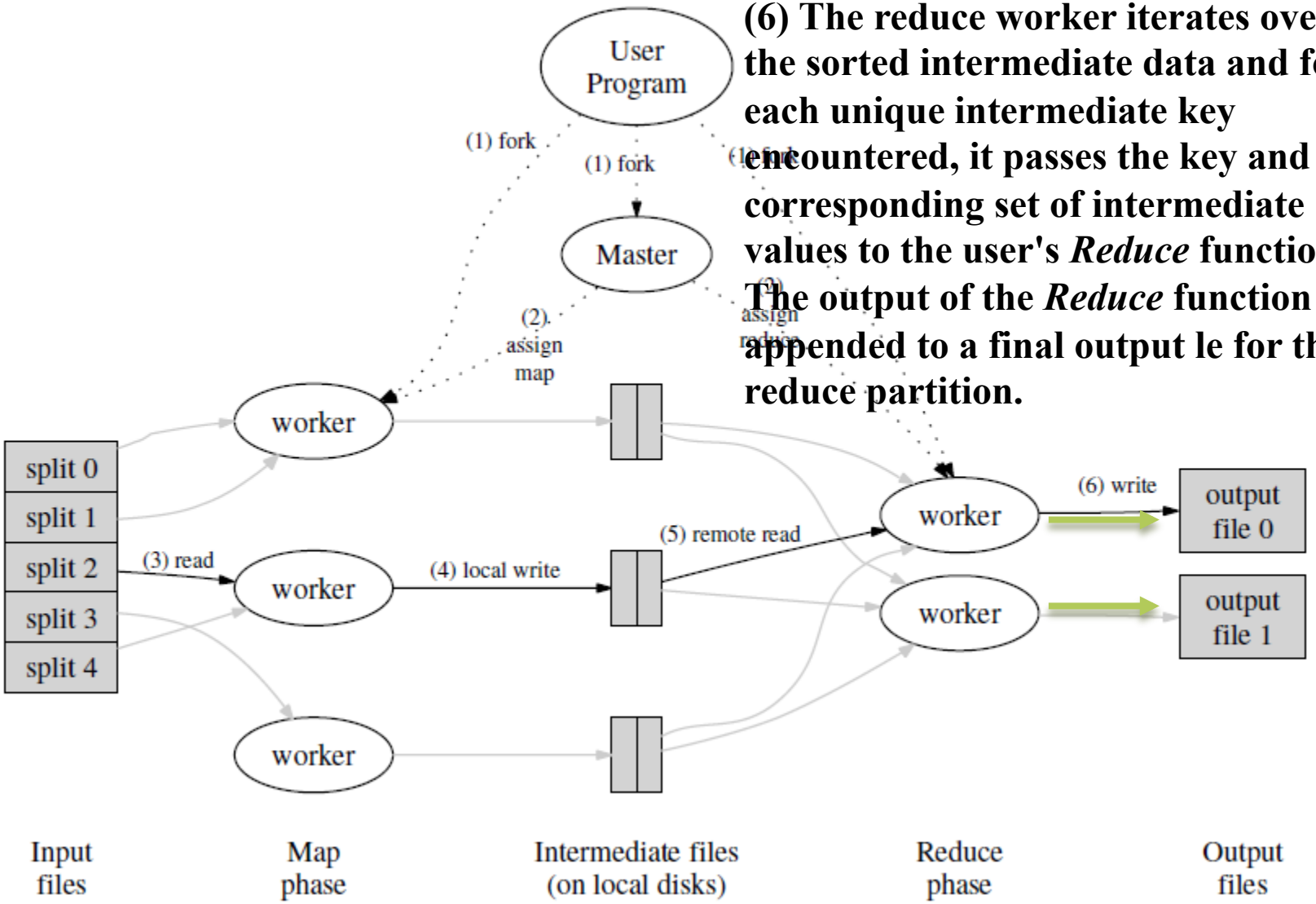


MapReduce的原理-7

(5) When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together.



MapReduce的原理-8



(6) The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's *Reduce* function. The output of the *Reduce* function is appended to a final output le for this reduce partition.

MapReduce的函数定义-1

- **More Examples**
  - **Distributed Grep:**

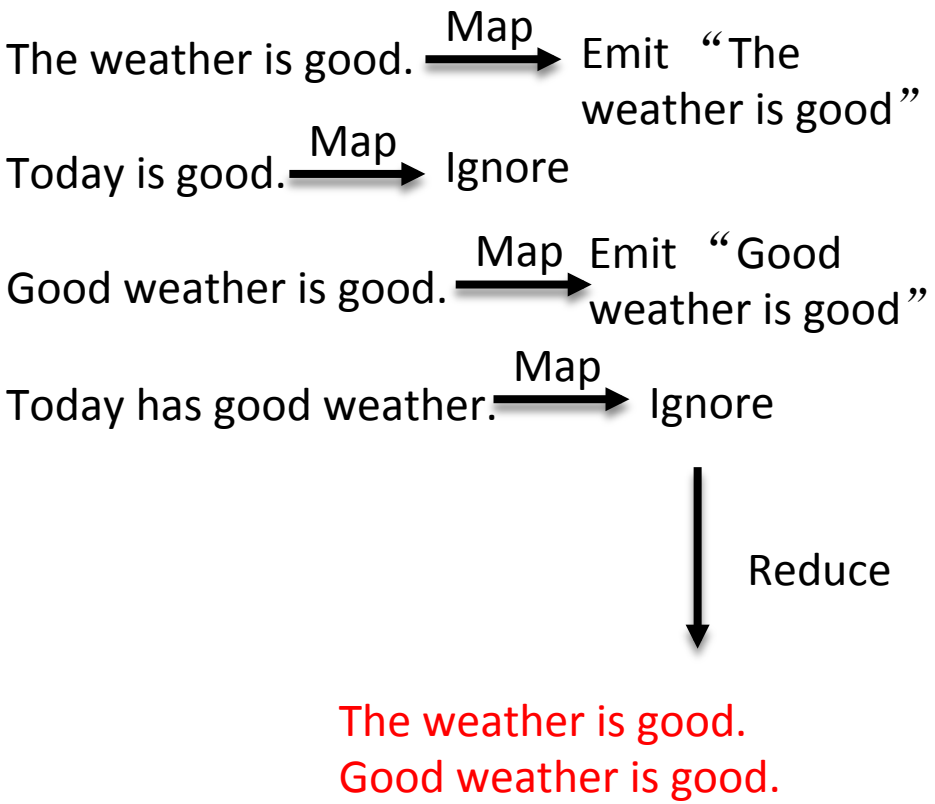
Map:

A line if it matches a supplied pattern.

Reduce:

An identity function that just copies the supplied intermediate data to the output.

If we want to know which sentences have the pattern 'weather is'.



MapReduce的函数定义-2

- **More Examples**
  - **Count of URL Access Frequency:**

Map:

Process logs of web page requests and outputs **<URL, 1>**.

Reduce:

Add together all values for the same URL and emits a **<URL, total count>** pair.

The messages below are much more simplified.
The real logs of web page will have much more information.

218.161.64.101-"GET /example/1/ HTTP/1.1"
218.161.64.101-"GET /test/1/ HTTP/1.1"
218.161.64.101-"GET /example/2/ HTTP/1.1"

Map →

</example/1/, 1>
</test/1/, 1>
</example/2/, 1>

218.161.64.101-"GET /example/1/ HTTP/1.1"
218.161.64.101-"GET /example/1/ HTTP/1.1"
218.161.64.101-"GET /example/1/ HTTP/1.1"

Reduce →

</example/1/, 3>

Reduce ↓

</example/1/, 4>
</test/1/, 1>
</example/2/, 1>

MapReduce的函数定义-3

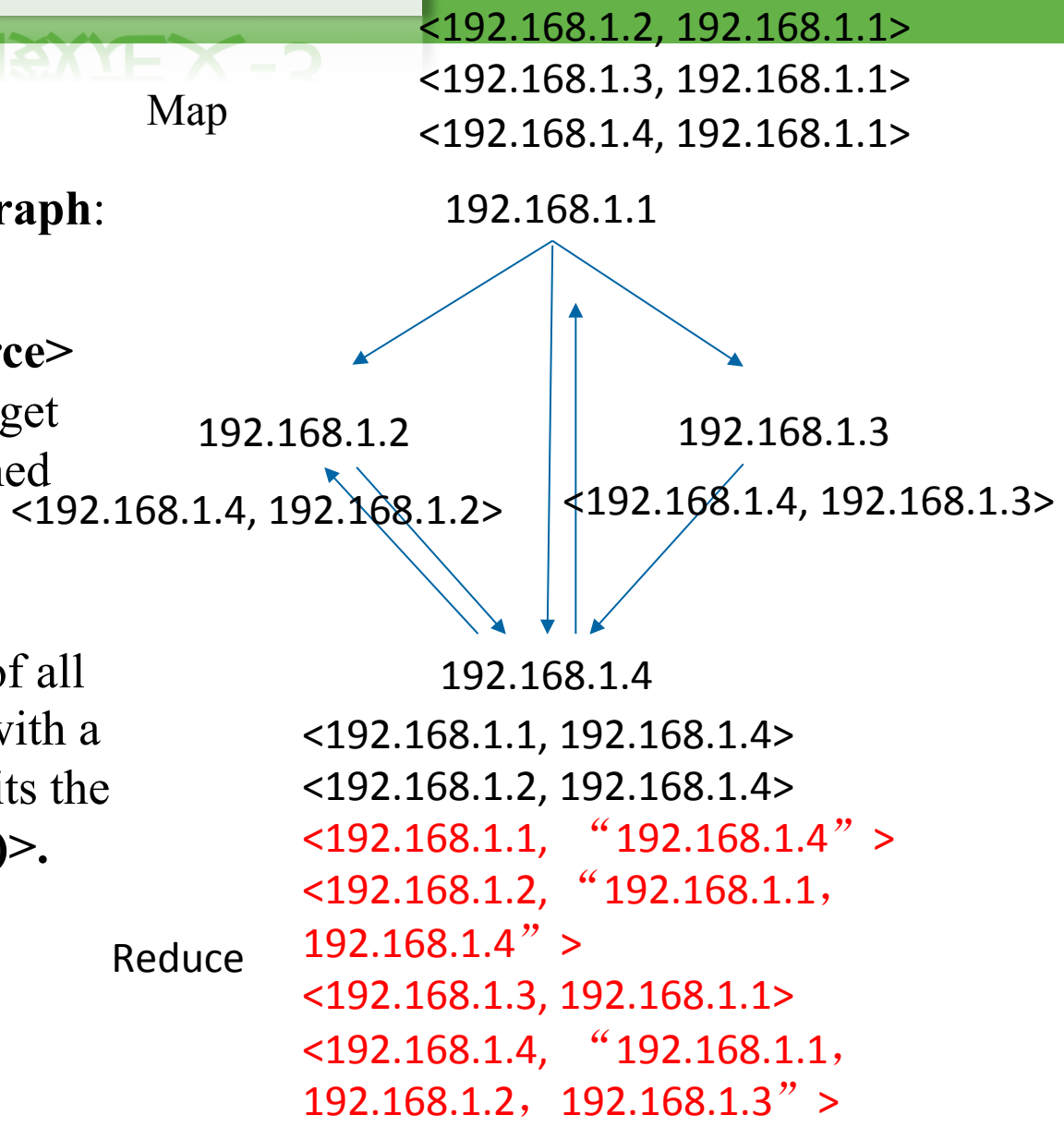
- **More Examples**
  - **Reverse Web-Link Graph**:

Map:

Output **<target, source>** pairs for each link to a target URL found in a page named source.

Reduce:

Concatenate the list of all source URLs associated with a given target URL and emits the pair**: <target, list(source)>.**



MapReduce的函数定义-4

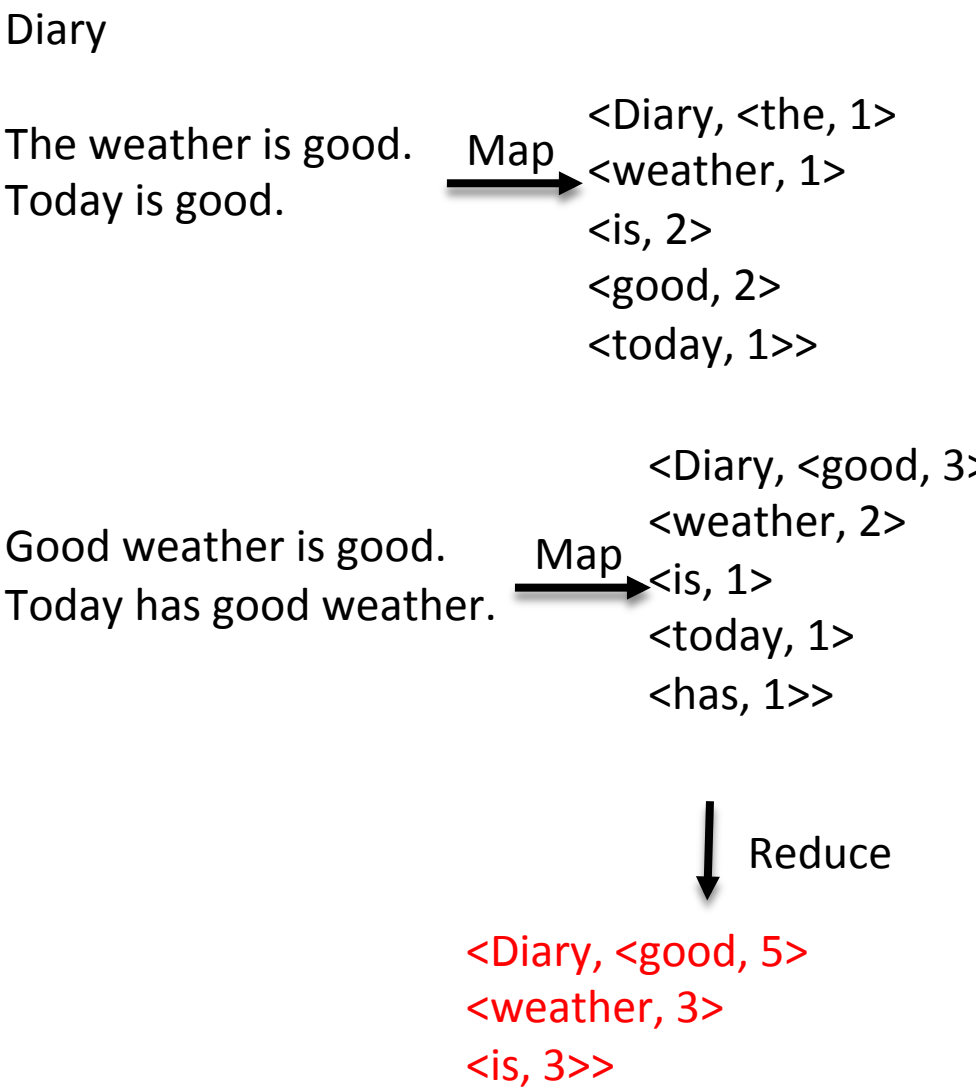
- **More Examples**
  - **Term-Vector per Host**: A term vector summarizes the most important words that occur in a document or a set of documents as a list of **<word; frequency>** pairs.

  Map:

  Emit a **<hostname, term vector>** pair for each input document.

  Reduce:

  Add these term vectors together, throwing away infrequent terms, and then emits a final **<hostname, term vector>** pair.



MapReduce的函数定义-5

- **More Examples**
  - **Inverted Index**:

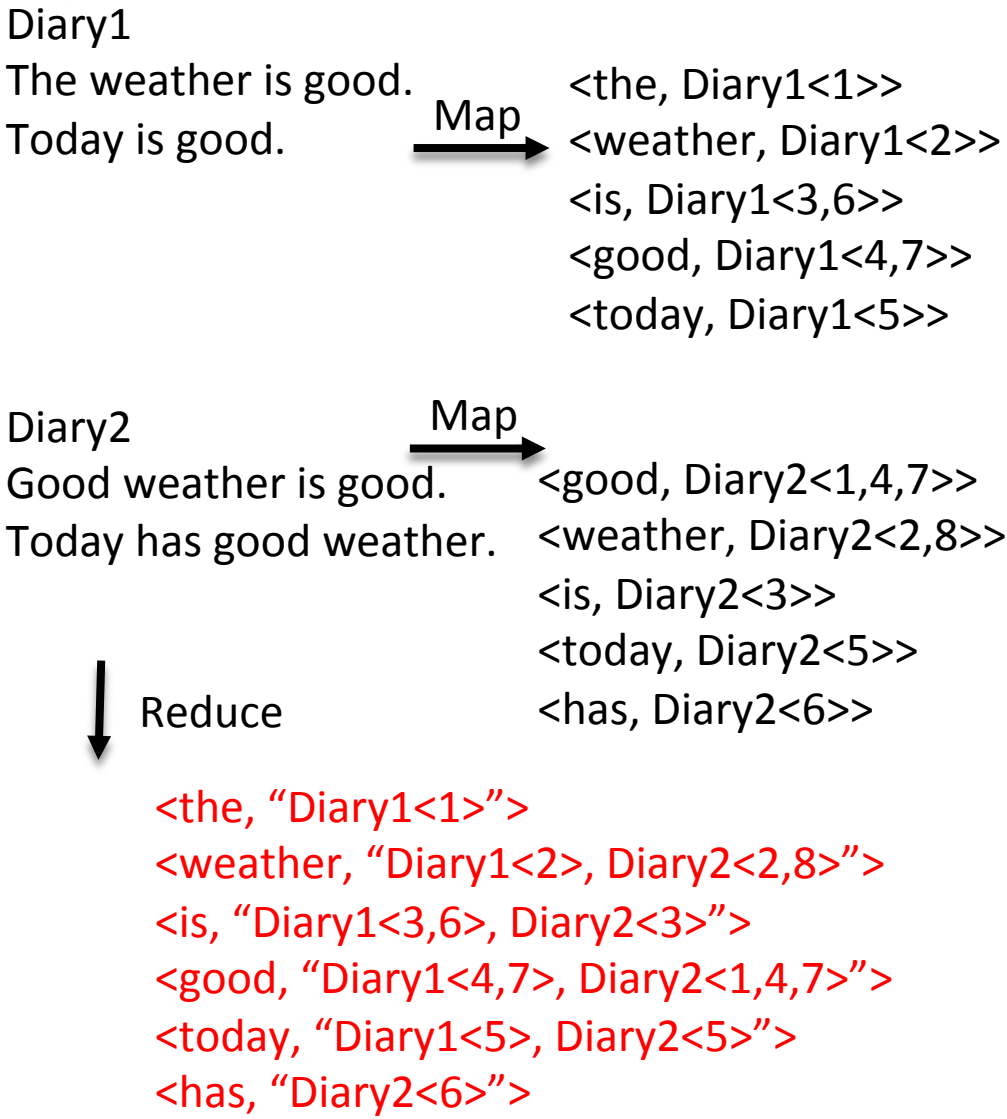
Map:

Parse each document, and emit a sequence of **<word, document ID>** pairs.

Reduce:

Accept all pairs for a given word, sorts the corresponding document IDs and emits a **<word, list(document ID)>** pair.

The set of all output pairs forms a simple inverted index. It is easy to augment this computation to keep track of word positions.



MapReduce的并发性

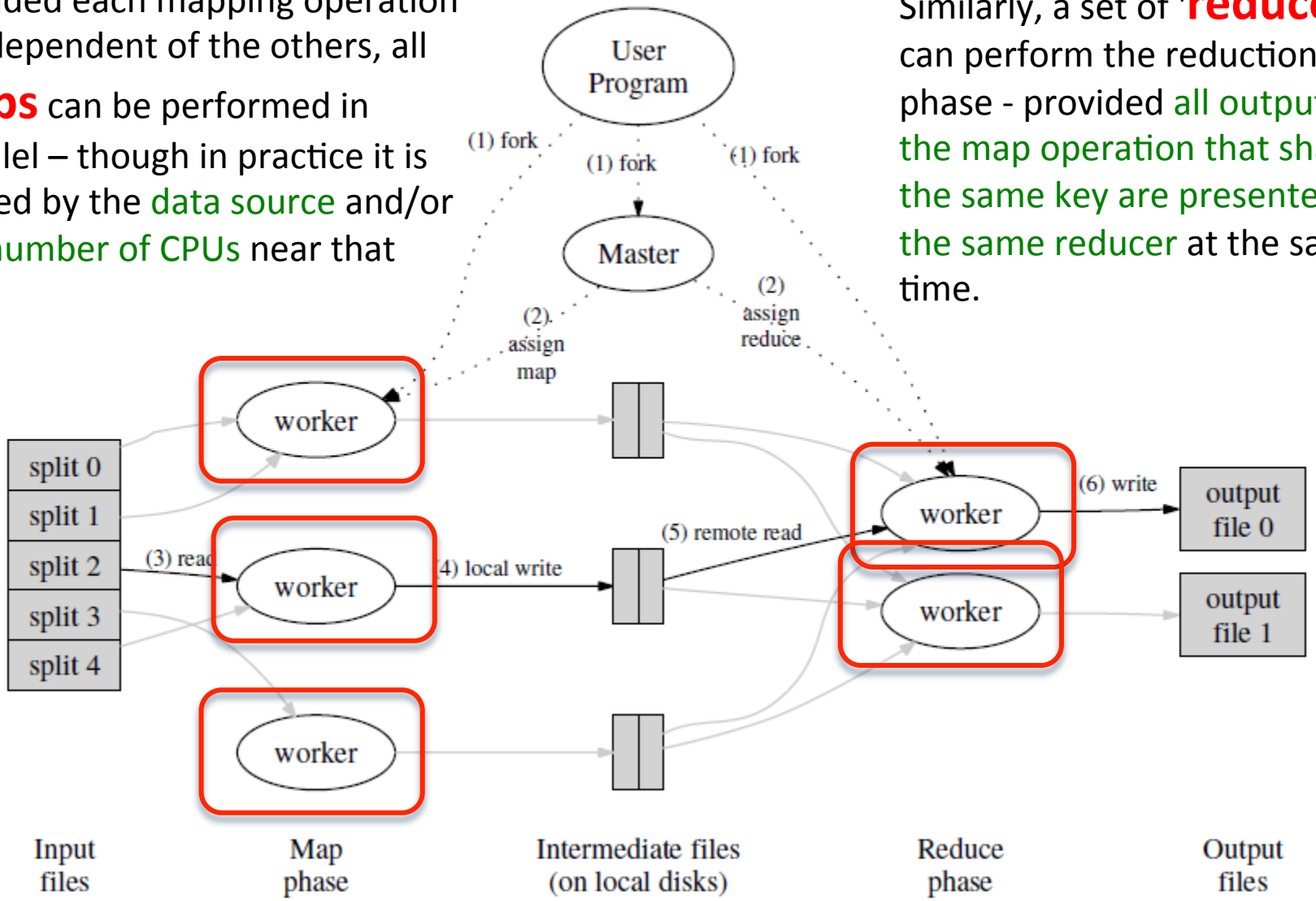
Now we can give a solution to the issues we mentioned.

- how to parallelize the computation
- distribute the data

- MapReduce allows for distributed processing of the map and reduction operations.
- While this process can often appear inefficient compared to algorithms that are more sequential, MapReduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use MapReduce to sort a petabyte of data in only a few hours.

MapReduce的并发性

Provided each mapping operation is independent of the others, all **maps** can be performed in parallel – though in practice it is limited by the data source and/or the number of CPUs near that data.

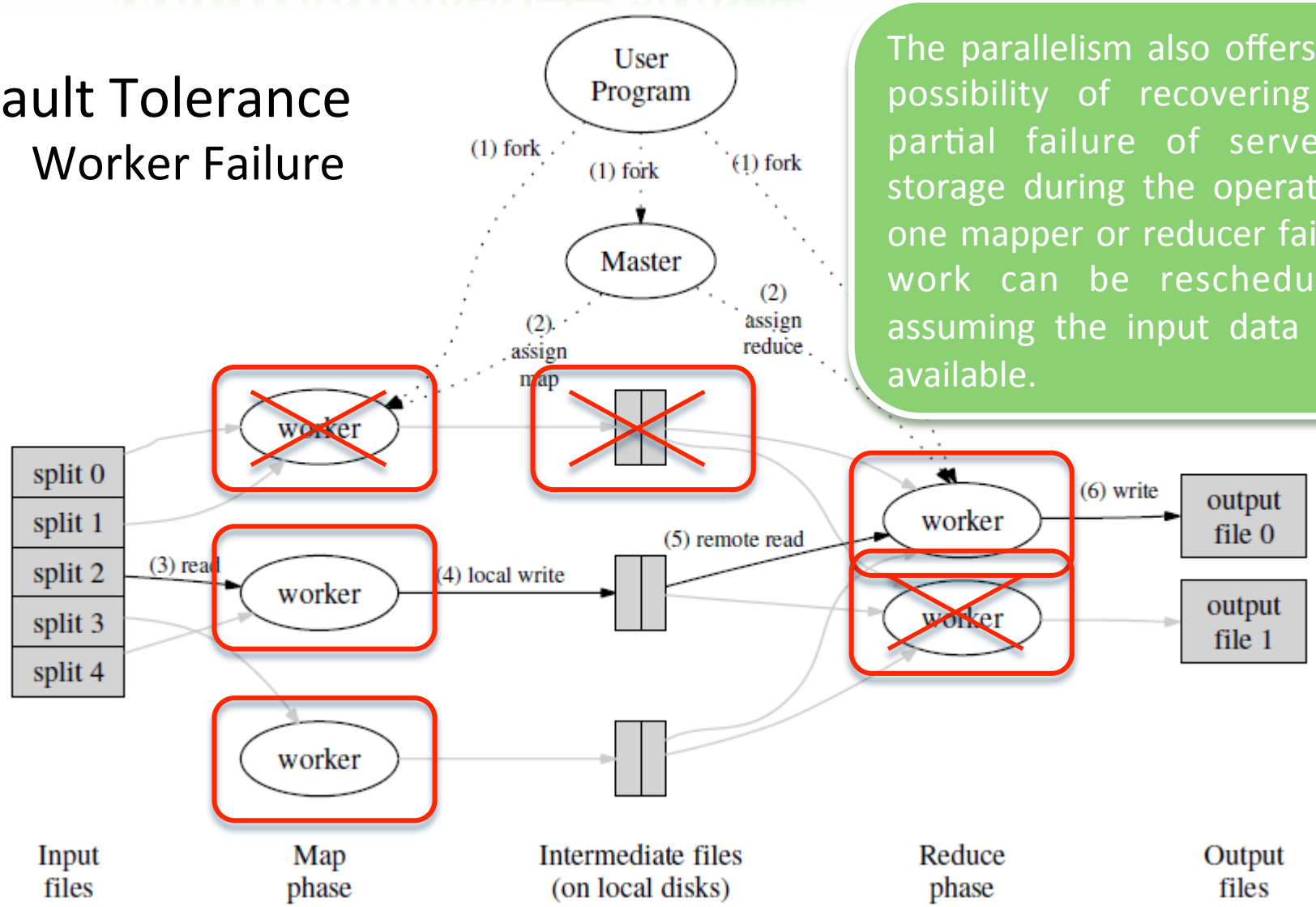


Similarly, a set of '**reducers**' can perform the reduction phase - provided all outputs of the map operation that share the same key are presented to the same reducer at the same time.

MapReduce的失效处理

Fault Tolerance

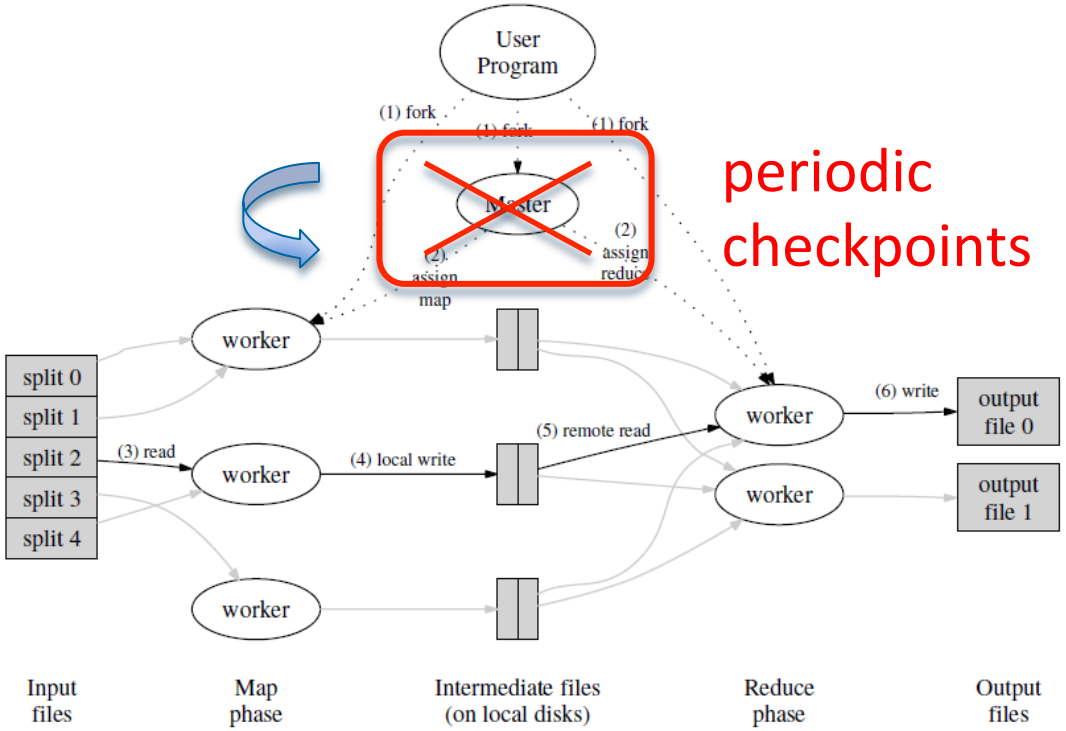
- Worker Failure



The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available.

MapReduce的性能

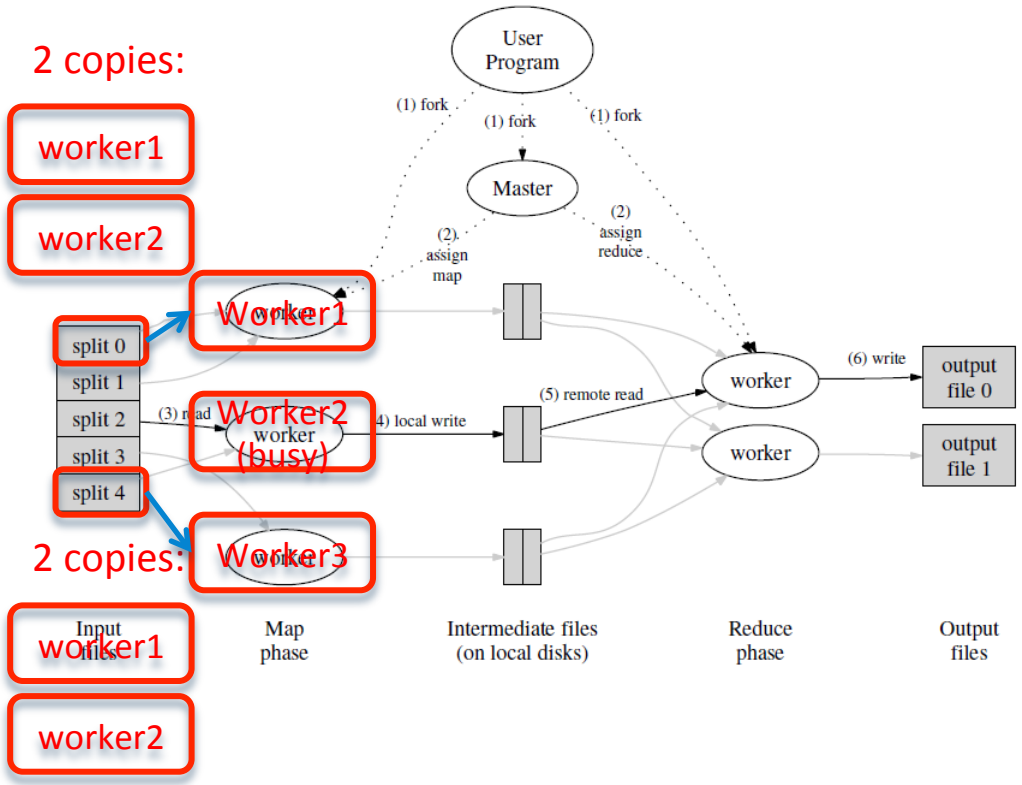
- Fault Tolerance
  - Worker Failure
  - Master Failure



periodic checkpoints

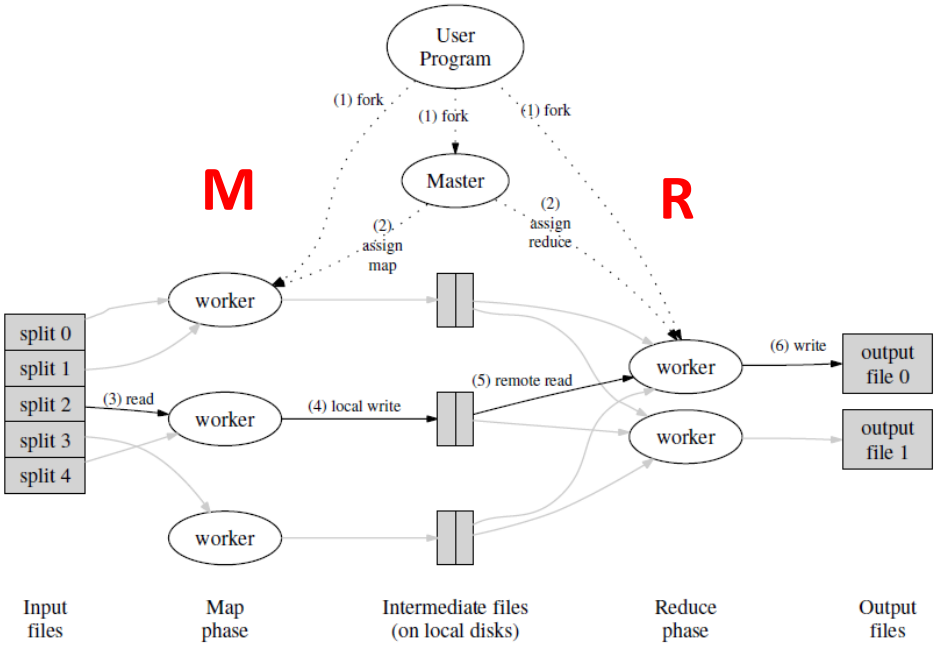
MapReduce的性能

- Locality
- The master takes the location information of the input files into account and attempts to schedule a map task on a machine that contains a replica of the corresponding input data.
- Failing that, it attempts to schedule a map task near a replica of that task's input data (e.g., on a worker machine that is on the same network switch as the machine containing the data).



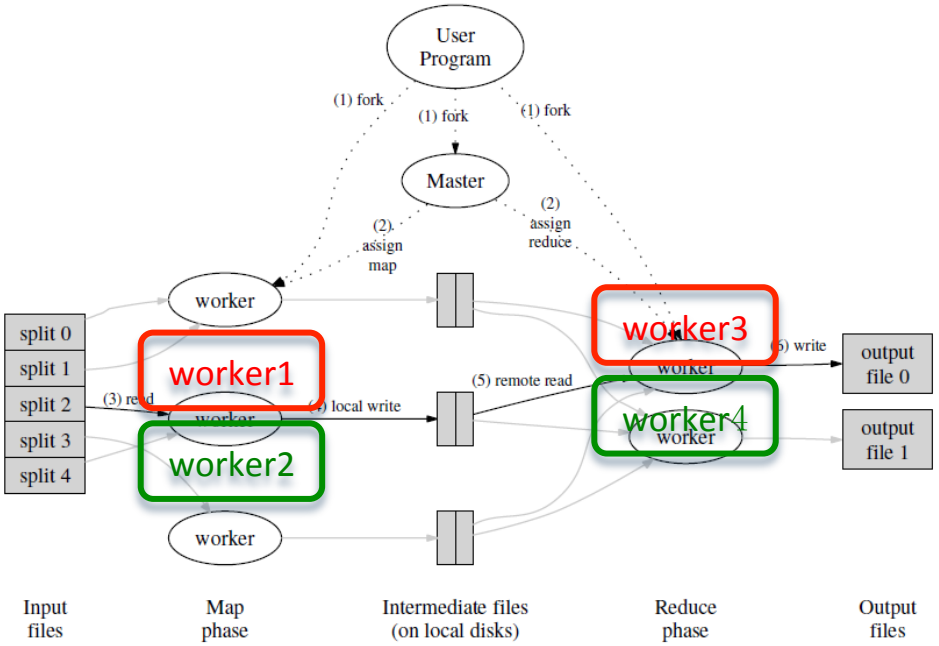
MapReduce的性能

- Task Granularity
- M and R should be much larger than the number of worker machines. Having each worker perform many different tasks improves dynamic load balancing, and also speeds up recovery when a worker fails.
- We often perform MapReduce computations with M = 200,000 and R = 5,000 , using 2,000 worker machines.



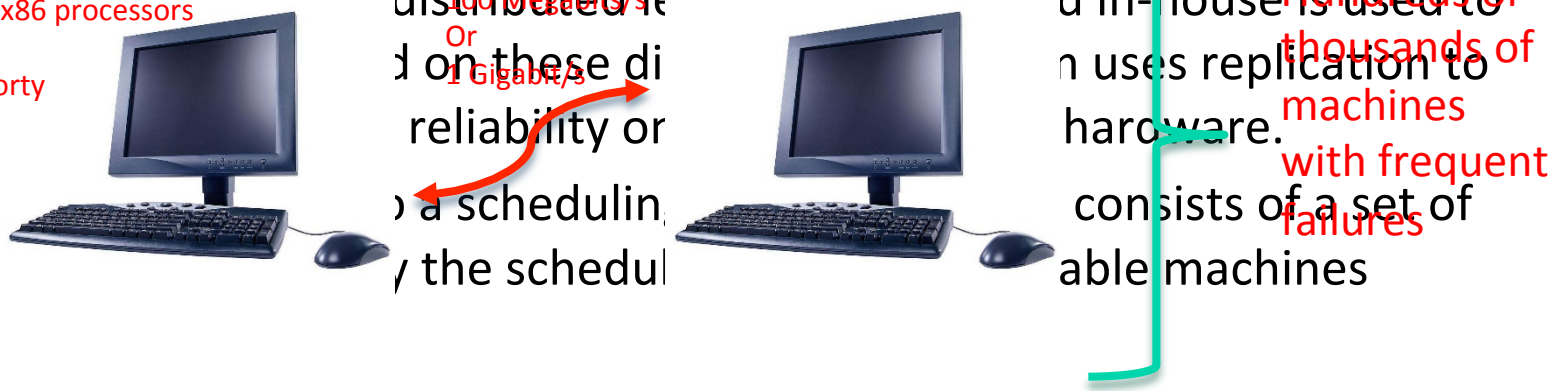
MapReduce的性能

- Backup Tasks
- One of the common causes that lengthens the total time taken for a MapReduce operation is a “straggler”: a machine that takes an unusually long time to complete one of the last few map or reduce tasks in the computation.
- When a MapReduce operation is close to completion, the master schedules backup executions of the remaining in-progress tasks. The task is marked as completed whenever either the primary or the backup execution completes.



MapReduce的实现

- (1) Machines are typically dual-processor x86 processors running Linux, with 2-4 GB of memory per machine.
- (2) Commodity networking hardware is used . Typically either 100 megabits/second or 1 gigabit/second at the machine level, but averaging considerably less in overall bisection bandwidth.
- (3) A cluster consists of hundreds or thousands of machines, and therefore machine failures are common.
- (4) Storage is provided by inexpensive IDE disks attached directly to indiv distributed le d in-house is used to mana d on these di n uses replication to provi reliability or hardware.
- (5) U a schedulin consists of a set of tasks y the schedul able machines withi



参考文献

[1] http://en.wikipedia.org/wiki/MapReduce

[2] Dean, Jeffrey & Ghemawat, Sanjay (2004). “MapReduce: Simplified Data Processing on Large Clusters”. OSDI, 2004