Distributed Batch Analytics with MapReduce

Prof. Pamod Bhatotia
https://dse.in.tum.de/bhatotia/

How much data?

processes 20 PB a day (2008)
crawls 20B web pages a day (2012)

>10 PB data, 75B DB calls per day (6/2012)

>100 PB of user data + 500 TB/day (8/2012)

S3: 449B objects, peak 290k request/second (7/2011)
1T objects (6/2012)

Distributed Systems!

Data-center

Cluster of 100s of thousands of machines

In today's class

How to easily write parallel applications on distributed computing systems?

1. MapReduce
2. Pig
   - A high-level language built on top of MapReduce

Design challenges

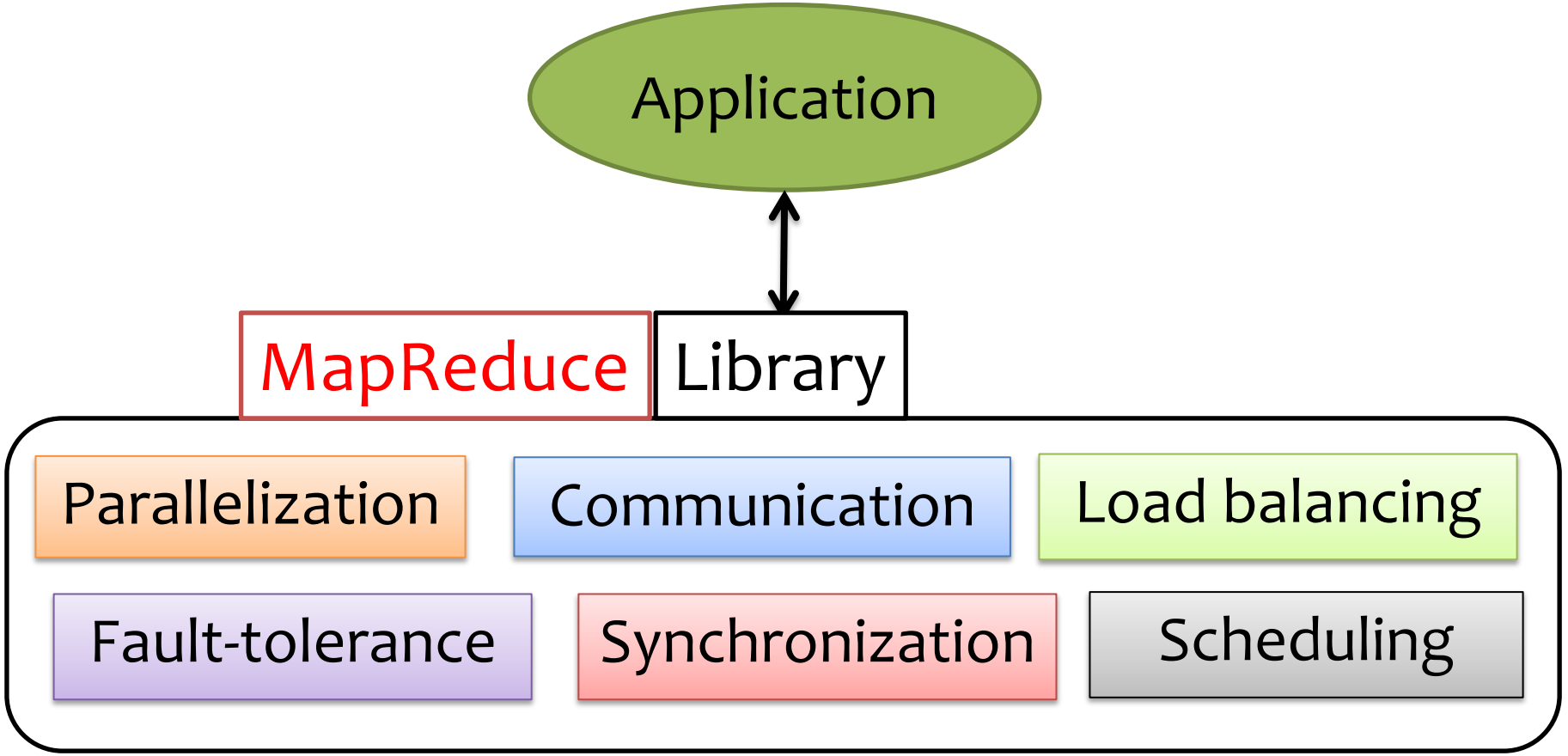
- How to parallelize application logic?
- How to communicate?
- How to synchronize?
- How to perform load balancing?
- How to handle faults?
- How to schedule jobs?

For each and every application!

Design | Implement | Optimize

Debug | Maintain

The power of abstraction



MapReduce

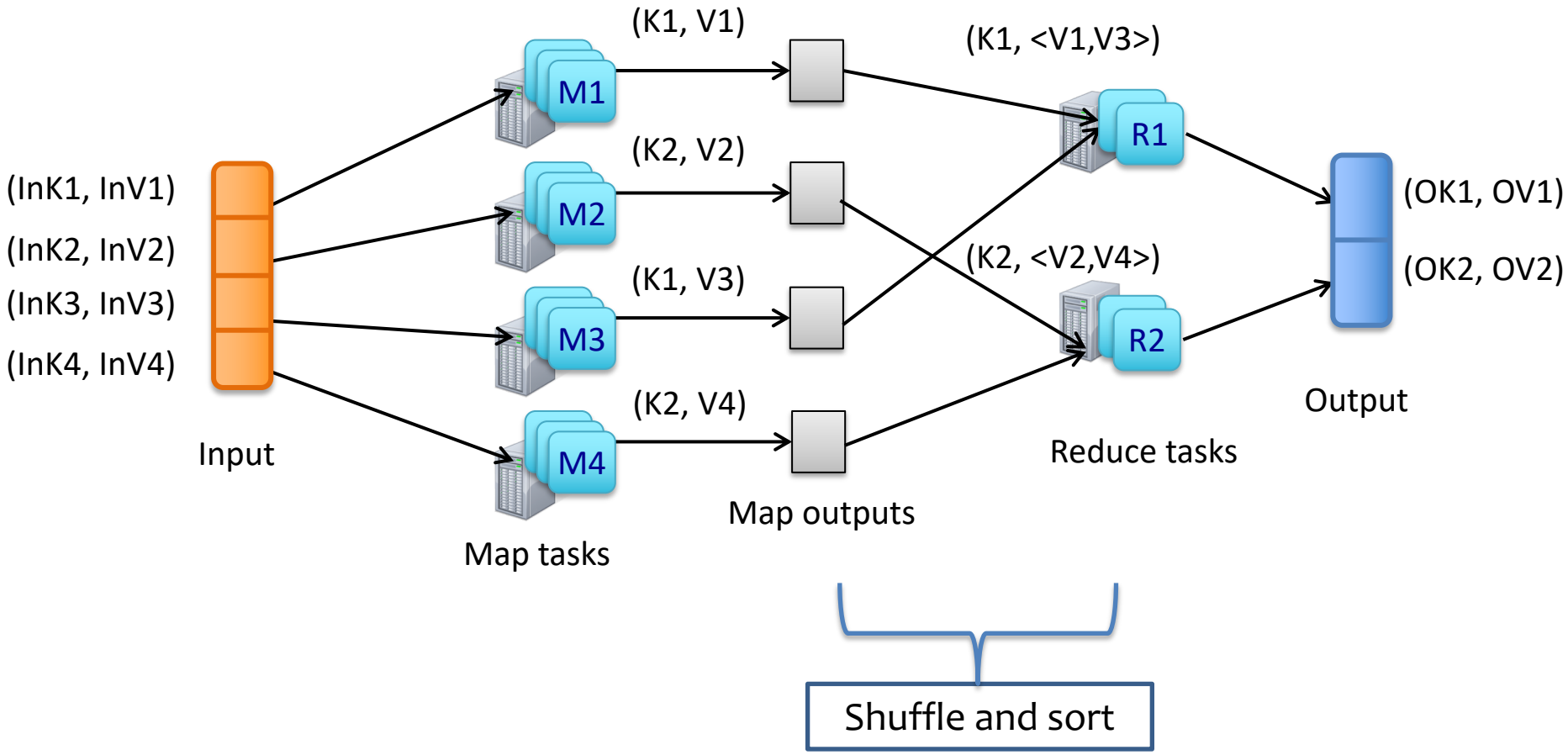
- Programming model
  - Programmer writes two methods: Map & Reduce
- Run-time library
  - Takes care of everything else!

MapReduce programming model

- Inspired from functional programming
  - Data-parallel application logic

- Programmer's interface:
  - Map(key, value) → (key, value)
  - Reduce(key, <value>) → (key,value)

MapReduce run-time system



An example: word-count

- Input:
  - Given a corpus of documents, such as Wikipedia
- Output:
  - Count the frequency of each distinct word

MapReduce for word-count

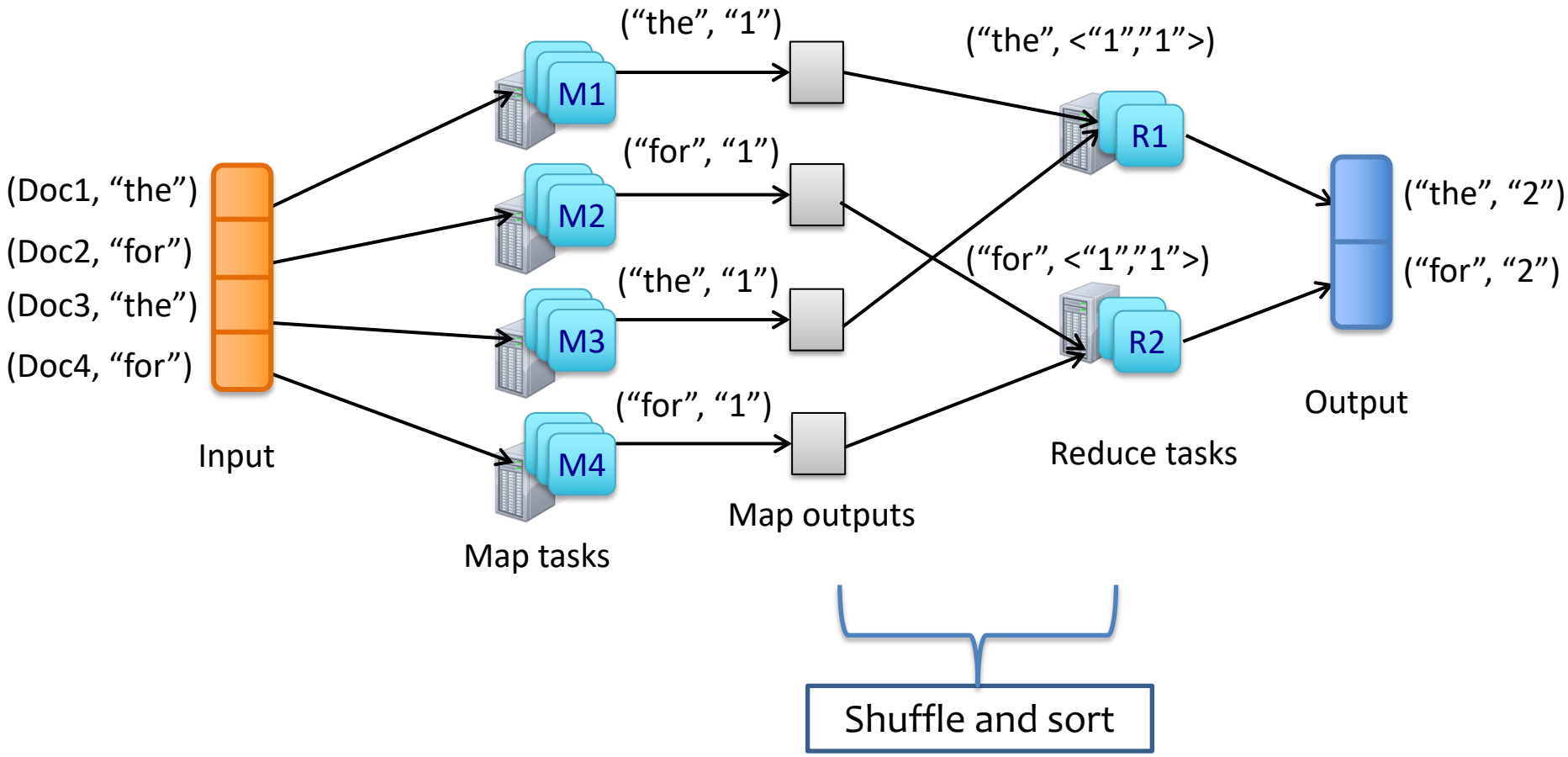
- map(string key, string value)

```
//key: document name
//value: document contents
for each word w in value
   EmitIntermediate(w, “1”);
```

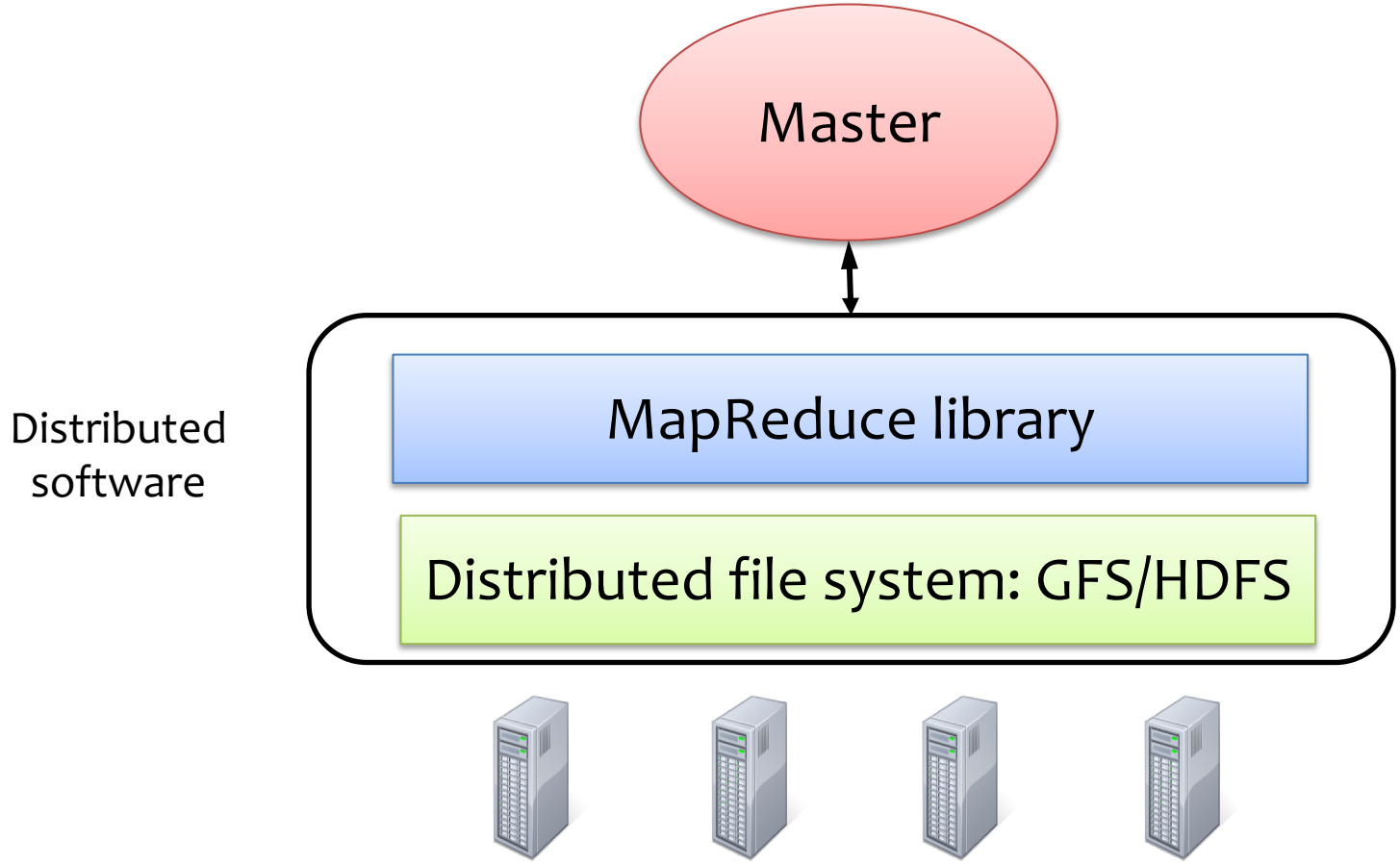
- reduce(string key, iterator values)

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

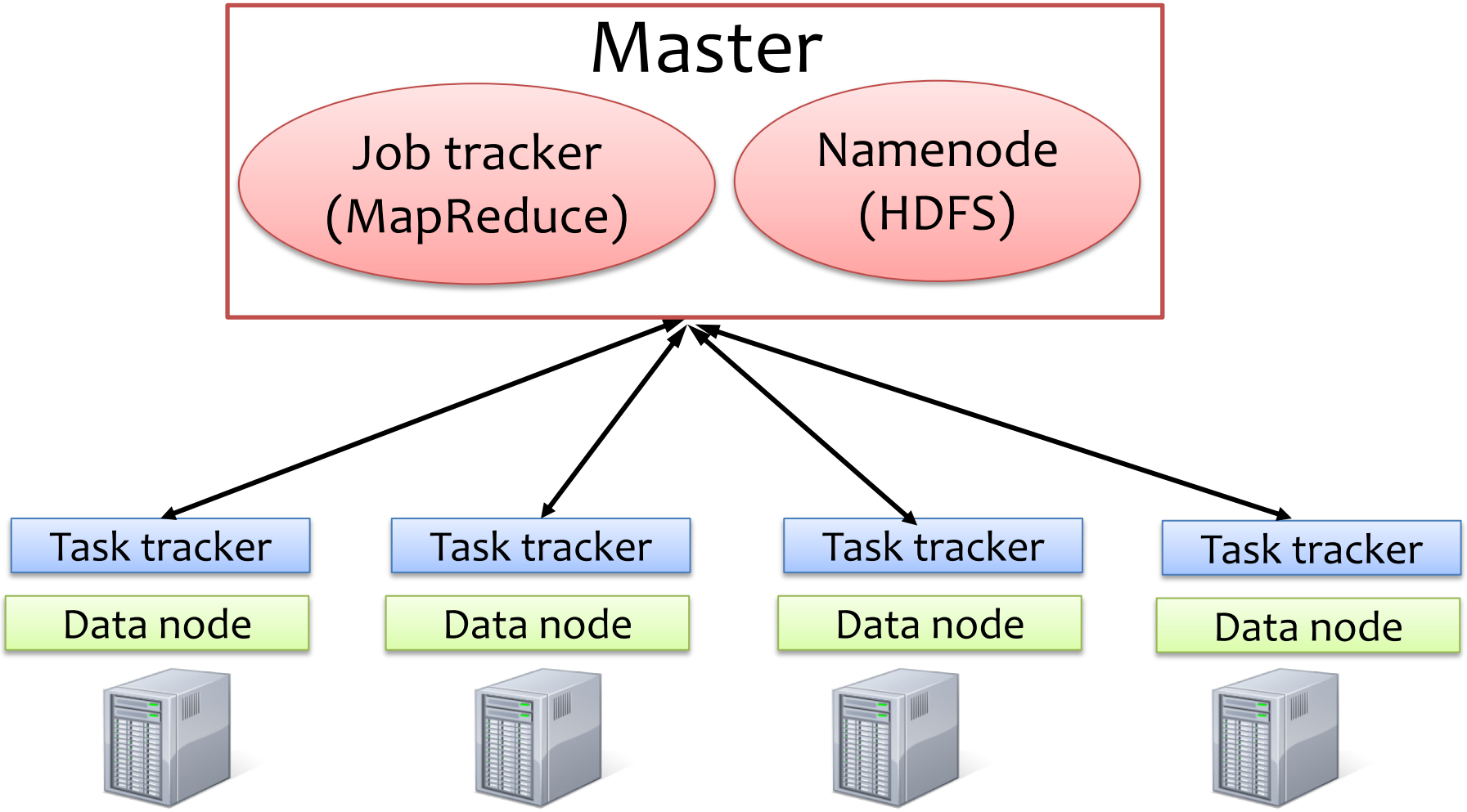
Word-count example



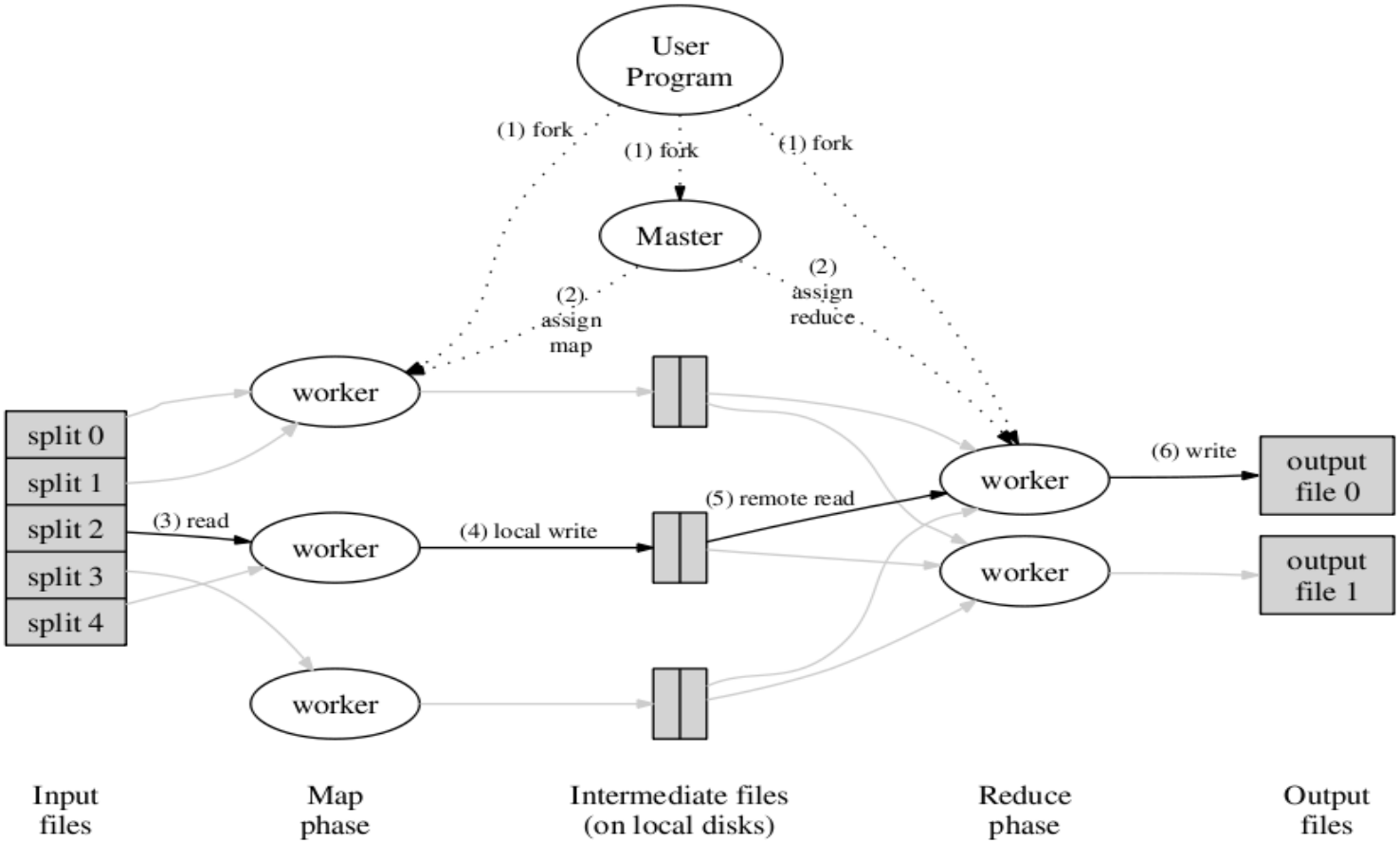
MapReduce software stack



MapReduce software stack



Runtime execution



MapReduce [OSDI'04]

Design challenges revisited

- Parallelization
- Communication
- Synchronization
- Load balancing
- Faults & semantics
- Scheduling

References

- Compulsory reading:
  - MapReduce [OSDI'04]: https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
- Recommended reading:
  - YARN [SoCC'13]
    - the next generation of M/R
  - Dryad [EuroSys'07]
    - Generalized framework for data-parallel computations
  - Spark [NSDI'12]
    - In-memory distributed data parallel computing