Charles Zheng

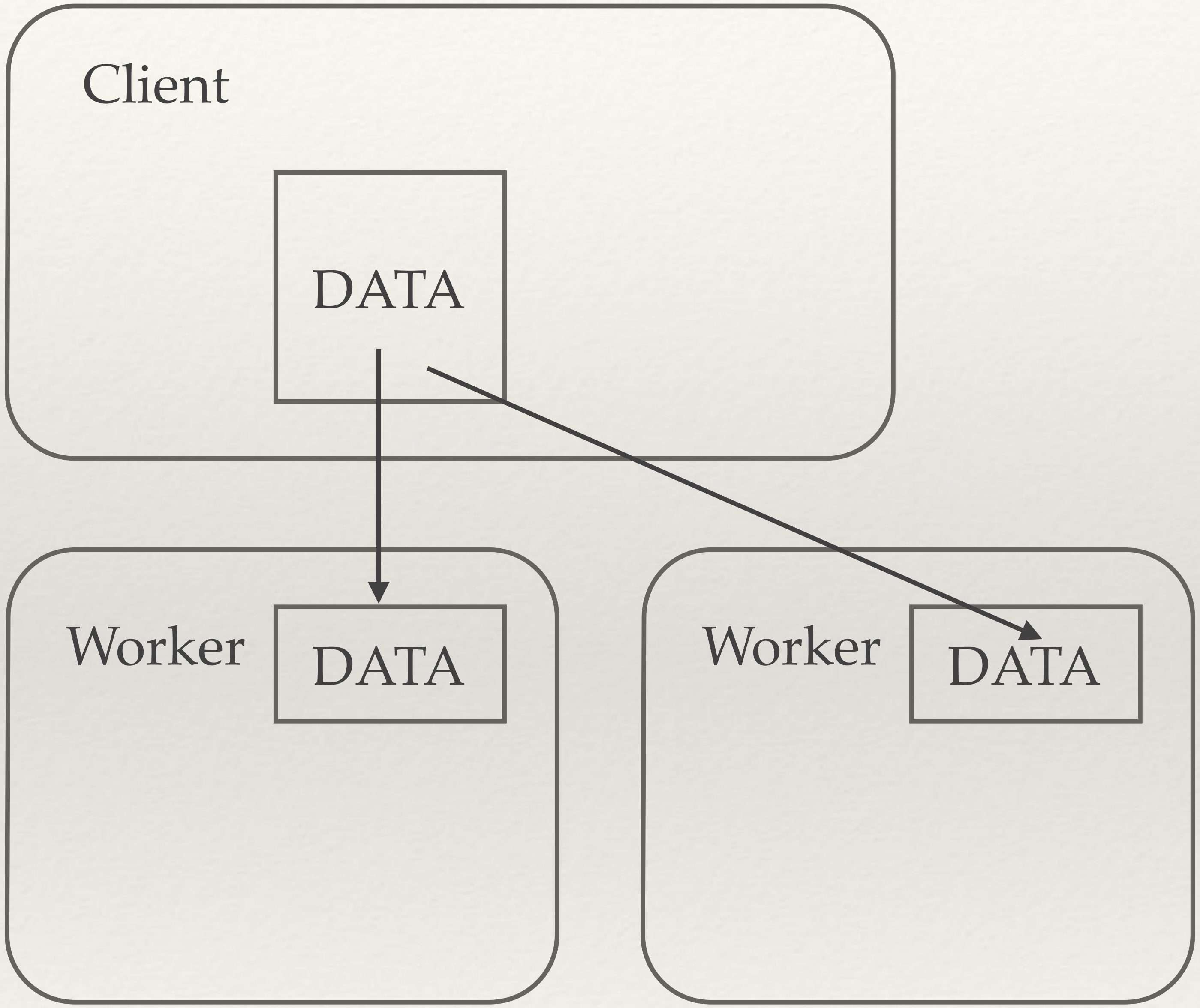
MapReduce with Apache Spark

Stats 290
March 11, 2015

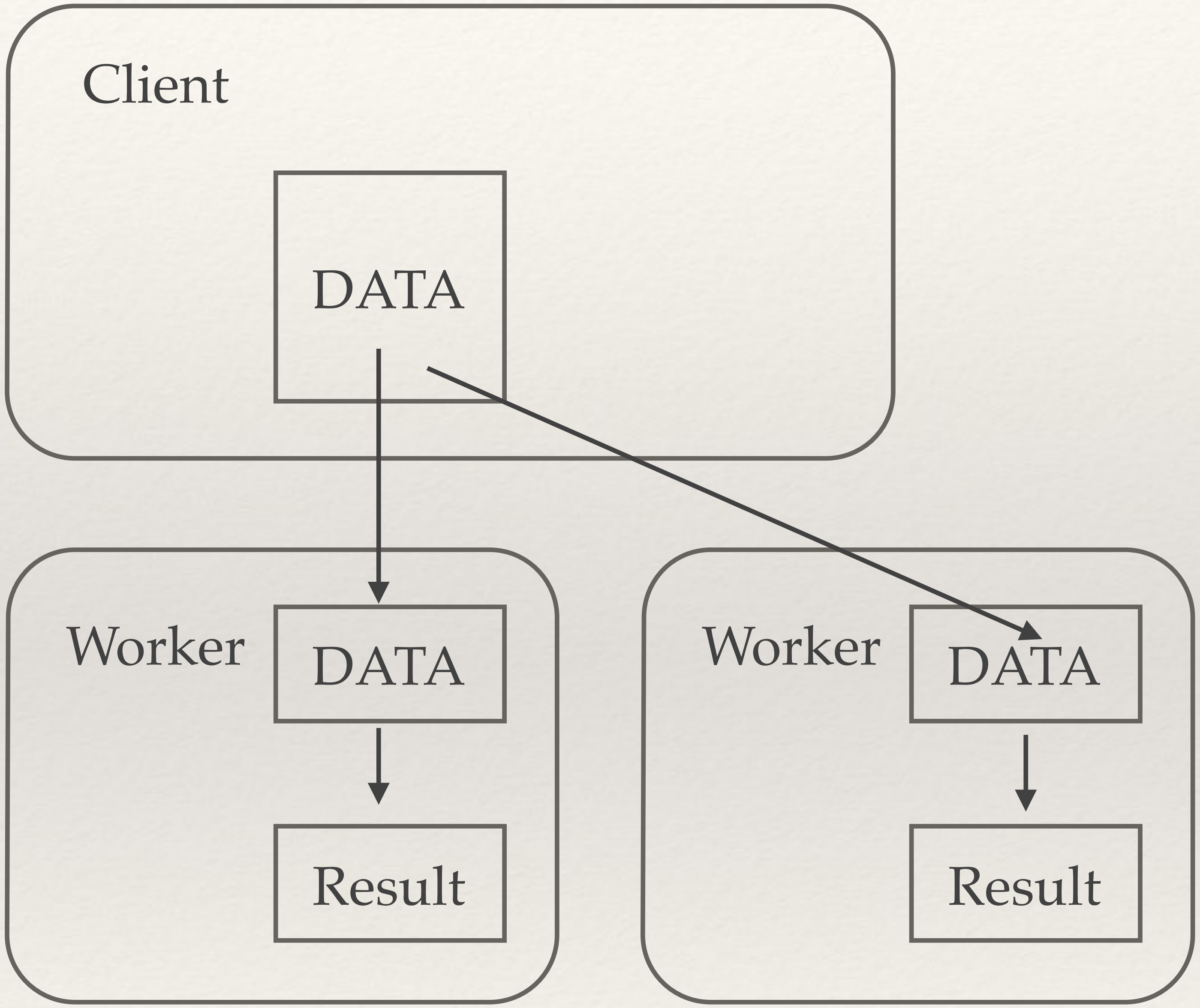
Review of Parallel/Distributed Computing

- Physical speed limit for sequential computing
- Only choice: parallelization
- Types of parallelization:
  - Multi-core on the same computer (`multicore`)
  - Simple network of workstations (`snow`)
  - Cluster computing (`batch`)

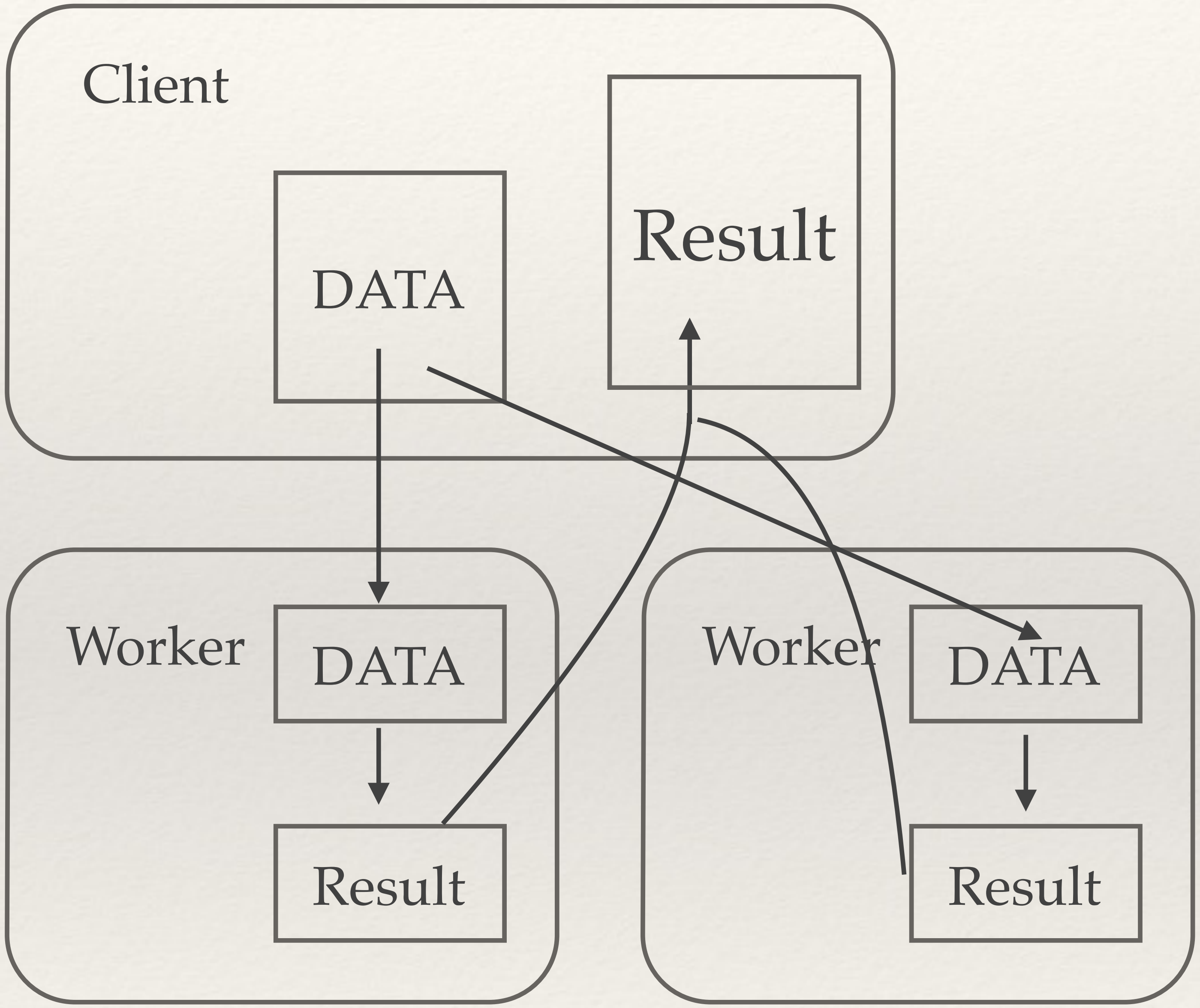
Typical Cluster Workflow



Typical Cluster Workflow



Typical Cluster Workflow



How could this be improved?

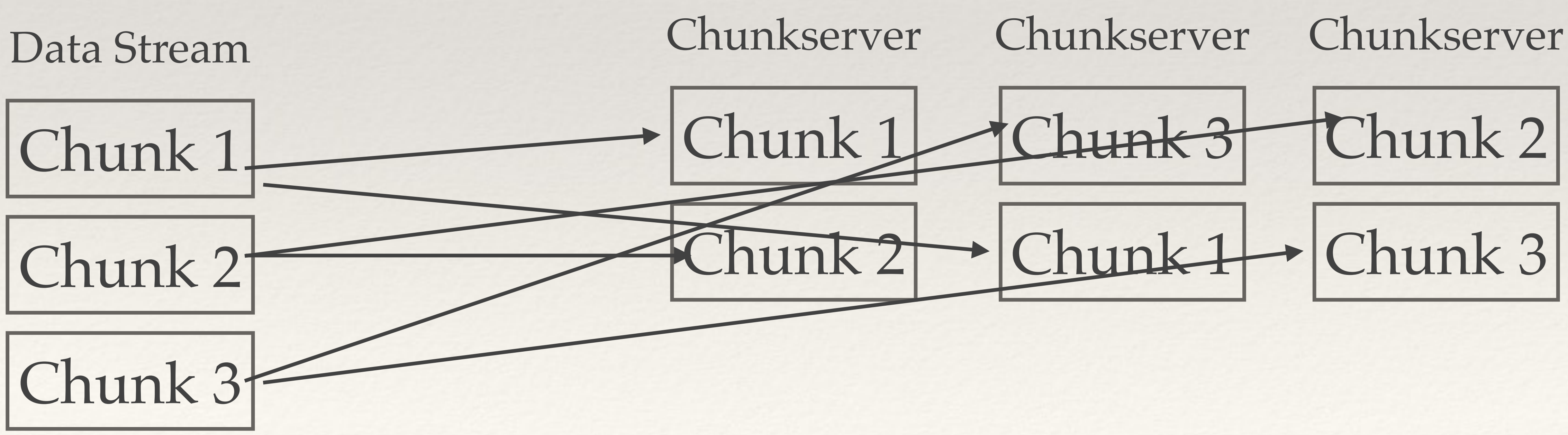
- Possible bottleneck if data / results have to be transferred to a central location
- What about tasks which require iterations?
  - Option 1. Aggregate results on a central node (slow)
  - Option 2. Use MPI (complicated)
- The basic mechanics of splitting, transferring and combining files could be abstracted

The MapReduce Framework

- A framework with two components:
- 1. A distributed file system
- 2. A “master node” which handles file splitting and assigning tasks to “worker nodes”
- Both the file system and computation are robust to individual machine failures

Distributed File System

- Files consist of many *records* (lines)
- Records are grouped into 64 MB *chunks*
- Multiple copies of each chunk spread across servers
- Master keeps track of addresses

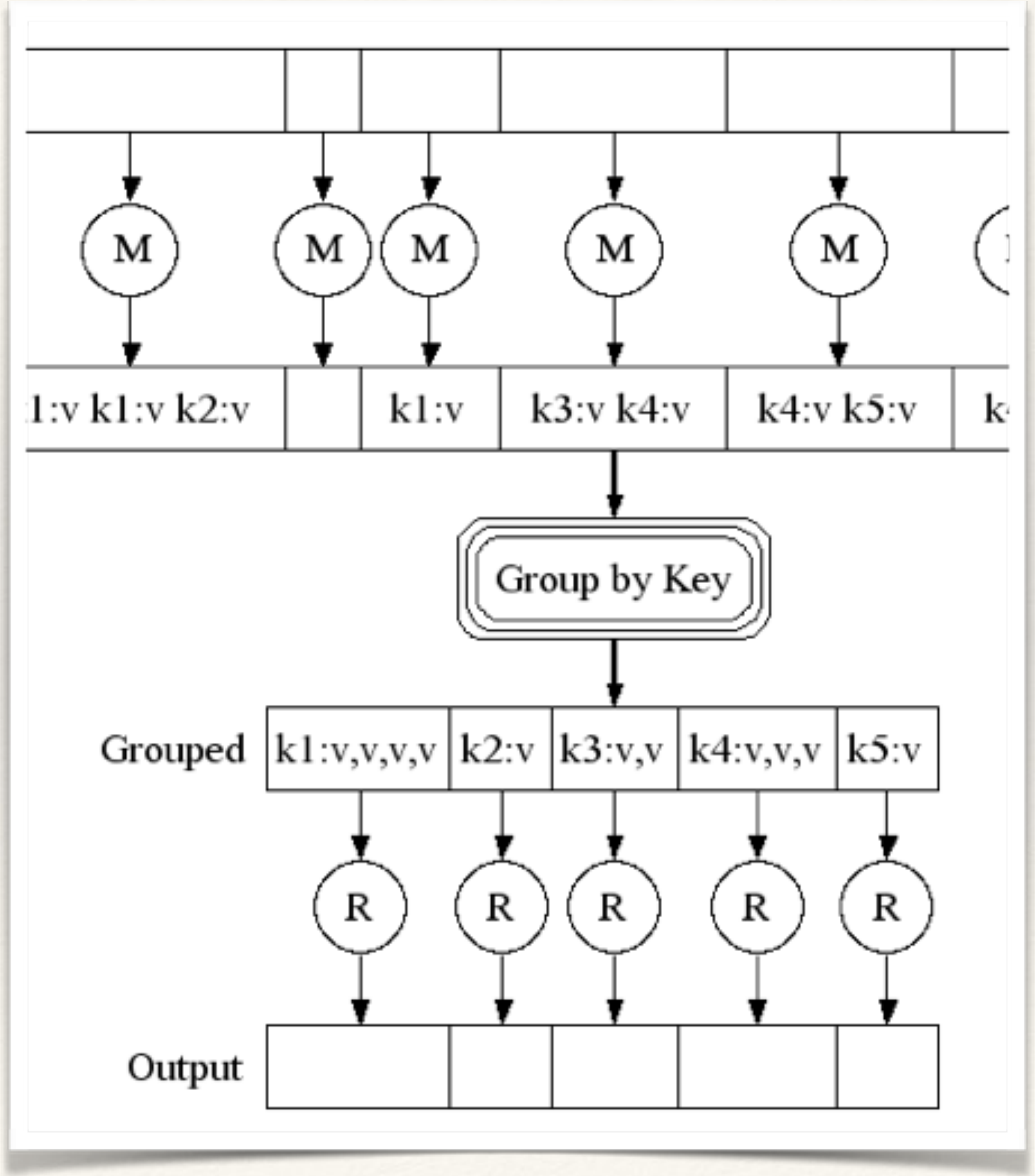


Computation: Map, Reduce

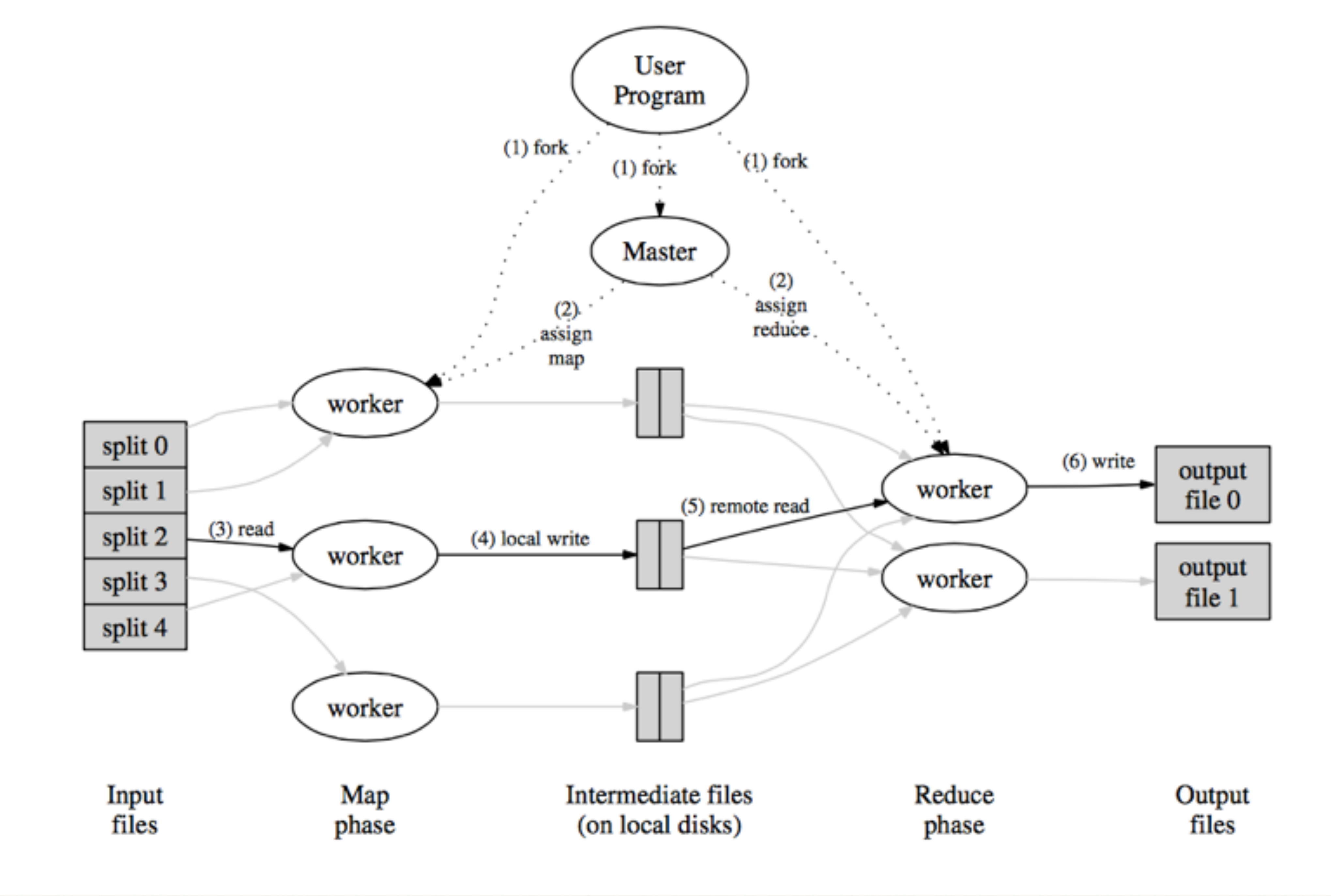
- Map (*FlatMap*): Apply the same operation to each record and produce key, value pairs
- Reduce (*Reduce by key*): Collect all values for a given key and aggregate them. Then write to file
- Master node assigns map or reduce task to workers
  - Mappers usually work on local chunks

Computation: Map, Reduce

- Map (*FlatMap*): Apply the same operation to each record and produce key, value pairs
- Reduce (*Reduce by key*): Collect all values for a given key and aggregate them. Then write to file
- Master node assigns map or reduce task to workers
  - Mappers usually work on local chunks



The MapReduce Framework

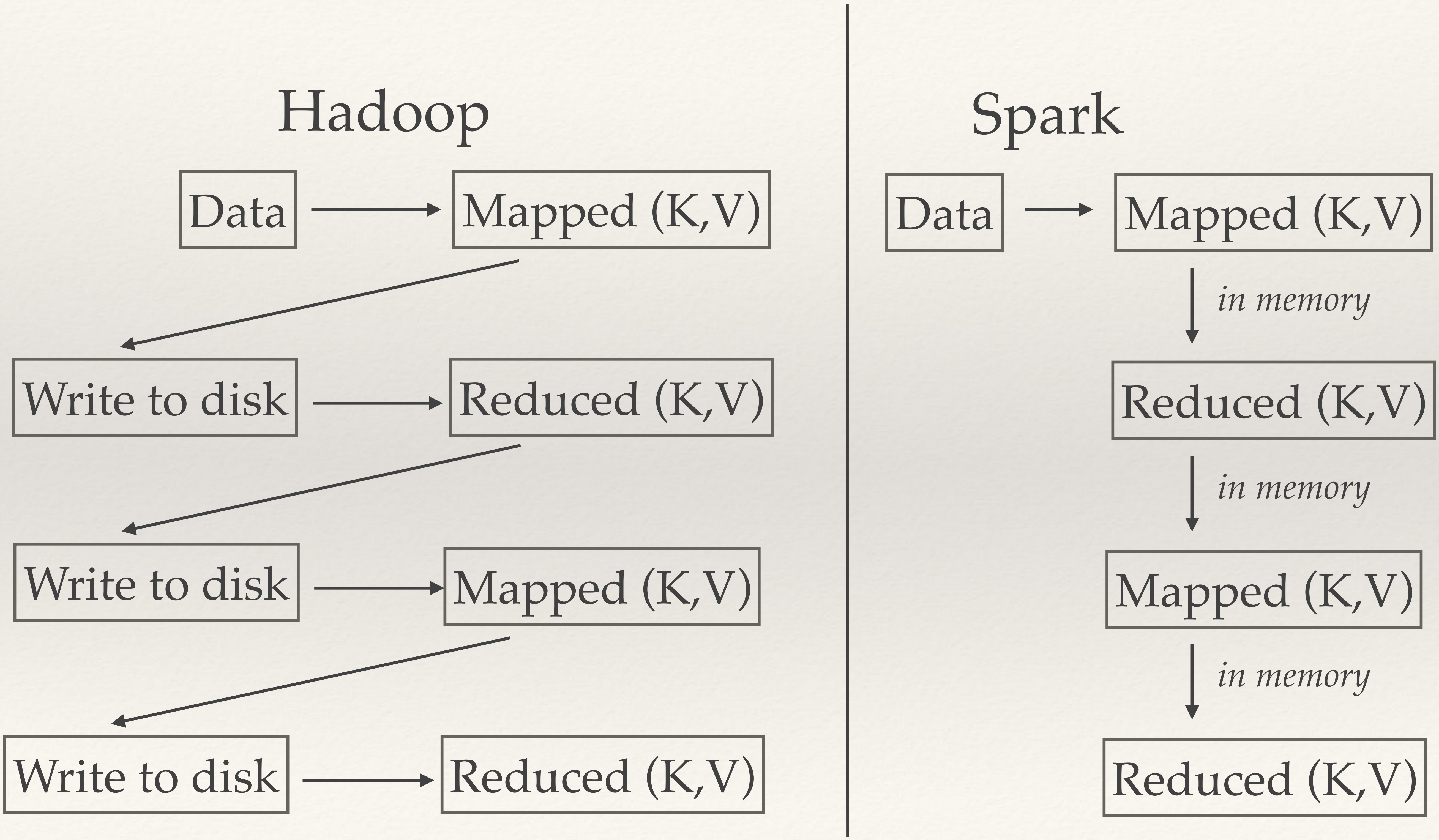


MapReduce Implementations

- Google in-house implementation
- Apache Hadoop
  - Based on Google's implementation
- Apache Spark
  - An extension of Hadoop
  - Optimized for iterative processes (10x speedup)

Spark Examples
(see html files)

Hadoop vs Spark



Advantages of Spark

- 10x faster for machine learning
- APIs for Java, Scala, Python, and R
- Speedup enables *interactive* exploration and analysis

Interfaces for Interactive Computing

- One option: Handle GUI on client side
  - E.g. Get results of computation from Rserve, then display in R
- Other option: Web sockets
  - Launch a web application from the cloud, then access it locally
  - RStudio server, IPython notebook, 0xdata (demos)

What can interactivity do for you?

- Adjust your experiments on the fly
  - Video: Spark streaming for neuroscience
- Scale up exploratory data analysis
- Probe for weaknesses in your methods using simulations
- [Your startup idea here]

Conclusions

- *MapReduce Framework*
  - Removes need to collect data in central node, and associated bottlenecks
  - More flexible than batch computing while remaining much simpler than MPI
- *Apache Spark implementation of MapReduce*
  - Faster iterations by using memory
  - Offers APIs in Java, Scala, Python and R

References

- Ghemawat, Gobioff, and Leung (Google). "The Google File System." *SOSP 2003*
- Dean and Ghemawat (Google). "MapReduce: Simplified Data Processing on Large Clusters." *OSDI 2004*
- Zaharia *et al* (UC Berkeley). "Spark: Cluster Computing with Working Sets." *HotCloud 2010*

Friday's lecture: Spark tutorial

- You will be given access to a Spark cluster on Amazon Elastic Compute Cloud
- Learn how to
  - use the Hadoop filesystem
  - launch IPython notebooks
  - run Spark jobs from your browser