MapReduce Model

Responsible:
Dr. Tegawendé F. BISSYANDE
tegawende.bissyande@uni.lu

Course Author:
Edson Ramiro Lucas Filho
Eduardo Cunha de Almeida

Teacher Assistant:
Médéric Hurier
mederic.hurier@uni.lu

Course Features

- Sep. 20th - **Introduction to Big Data**

Part 1. Databases and Query Models for Big Data

- Sep. 27th - **Relational Databases: Reminders**
- Oct. 4th - **Relational Databases: Internals**
- Oct. 11th - **NoSQL & NewSQL Databases**
- Oct. 18th - **MapReduce Model**
- Oct. 25th - **Hadoop and Spark**
- Nov. 8th - **Datalog Model**

Part 2. Data Analysis and Machine Learning

- Nov. 15th - **Statistics and Probabilities**
- Nov. 22th - **Communication and Visualization**
- Nov. 29th - **Features Engineering and Supervised Learning**
- Dec. 5st - **Unsupervised and Reinforcement Learning**
- Dec. 12th - **Homework Time**

What you should know after this class ?

- What is MapReduce and how it works.
- How we can scale with MapReduce
- How we can build a DBMS on top

Section Features

- Introduction
- MapReduce
- How does MapReduce Scale ?
- How do we build DBMSs on top of MapReduce ?

Introduction

What is scalability?

A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a scalable system.

Poor scalability can result in poor system performance, necessitating the reengineering or duplication of systems.

Database Systems for Advanced Applications: 16th International Conference, DASFAA 2011, Hong Kong, China, April 22-25, 2011, Proceedings. Yu, J.X. and Kim, M.H. and Unland, R.
http://books.google.lu/books?id=zKHUaaT0Oq0C

Characteristics of scalability and their impact on performance.
André B. Bondi. AT&T Labs
http://dl.acm.org/citation.cfm?doid=350391.350432

MapReduce

What is MapReduce ?

It's a programming model

to simplify distributed data processing

What is MapReduce ?

Based on functional programming paradigm

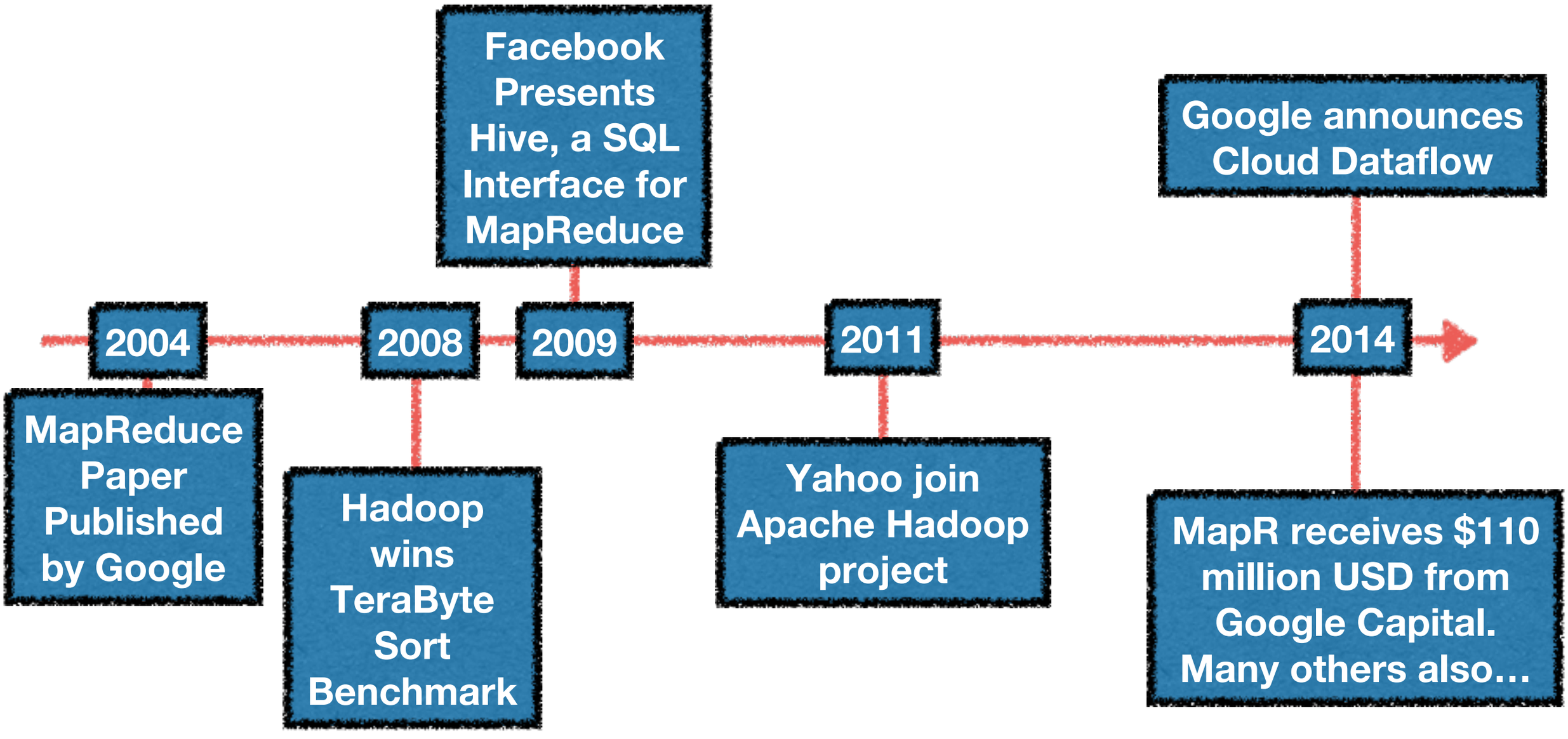
It's a programming model

to simplify distributed data processing

Hide from the user all problems related to distributed computation

Allows distributed processing of large data sets across clusters of thousands of machines

History

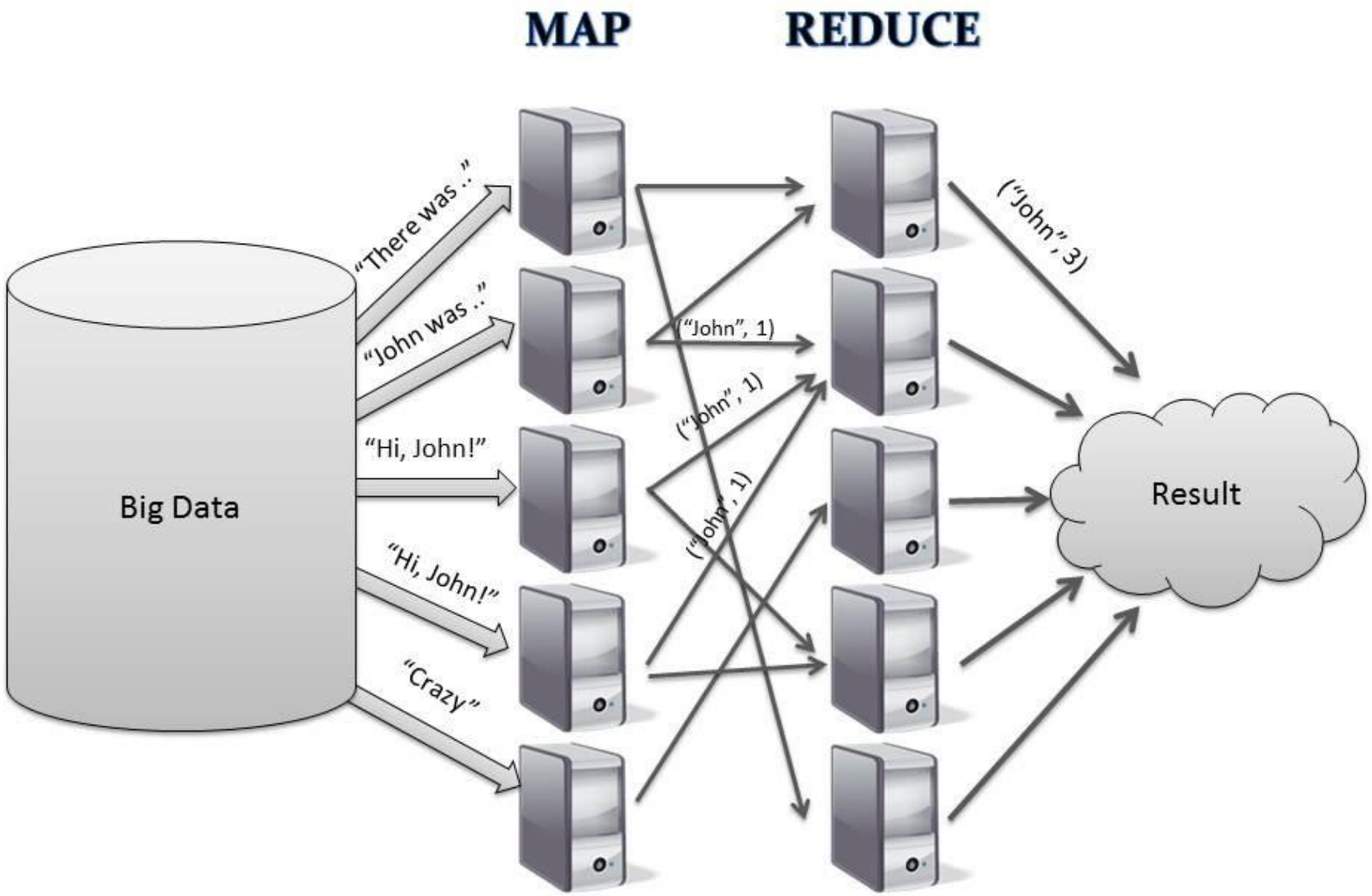

Facebook Presents Hive, a SQL Interface for MapReduce
Google announces Cloud Dataflow
2004
2008
2009
2011
2014
MapReduce Paper Published by Google
Hadoop wins TeraByte Sort Benchmark
Yahoo join Apache Hadoop project
MapR receives $110 million USD from Google Capital. Many others also…

MapReduce Implementations

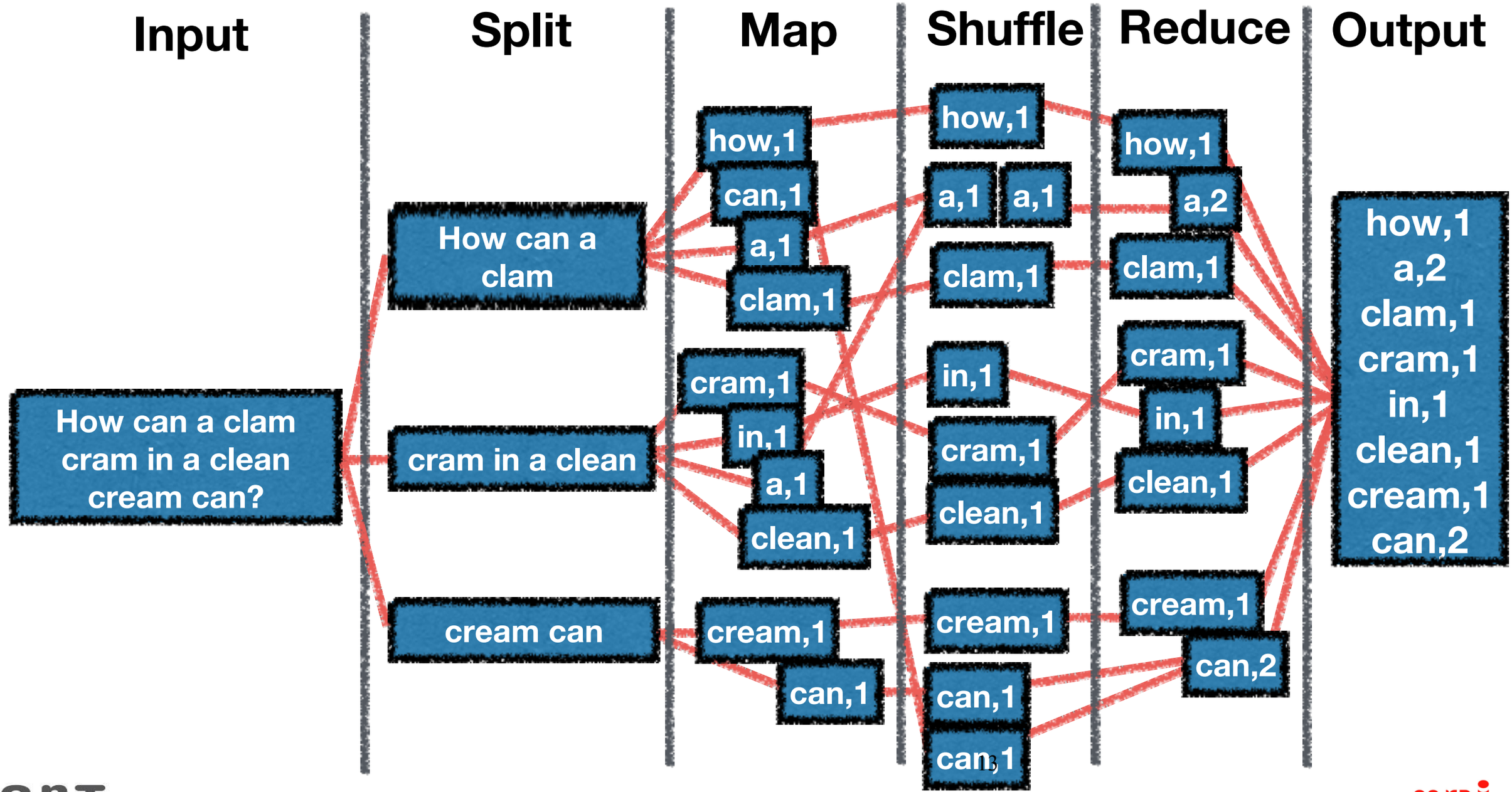
Owner	Name	Source Code
Microsoft	**Dryad**	https://github.com/MicrosoftResearch/Dryad
Apache	**Hadoop**	https://github.com/apache/hadoop
AMPLab (UC Berkley)	**Spark**	https://github.com/apache/spark

and many others…

Logical Model
The WordCount Example



Logical Model
The WordCount Example



MapReduce Functions

Based on functional programming paradigm

It's a programming model

Map(), Reduce(), Shuffle()...

MapReduce Pseudo-Code

```
function map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    emit (w, 1)

function reduce(String word, Iterator partialCounts):
  // word: a word
  // partialCounts: a list of aggregated partial counts
  sum = 0
  for each pc in partialCounts:
    sum += ParseInt(pc)
  emit (word, sum)
```

MapReduce Code - Mapper

```
class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    IntWritable        = new IntWritable(1);
    Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word,       );
        }
    }
}
```

MapReduce Code - Reducer

```
class Reduce extends Reducer
    <Text, IntWritable, Text, IntWritable> {

    public void reduce
        (Text key, Iterator<IntWritable> values, Context context){
        int sum = 0;
        while (values.hasNext()) {
                += values.next().get();
        }
        context.write(key, new IntWritable(       ));
    }

}
```

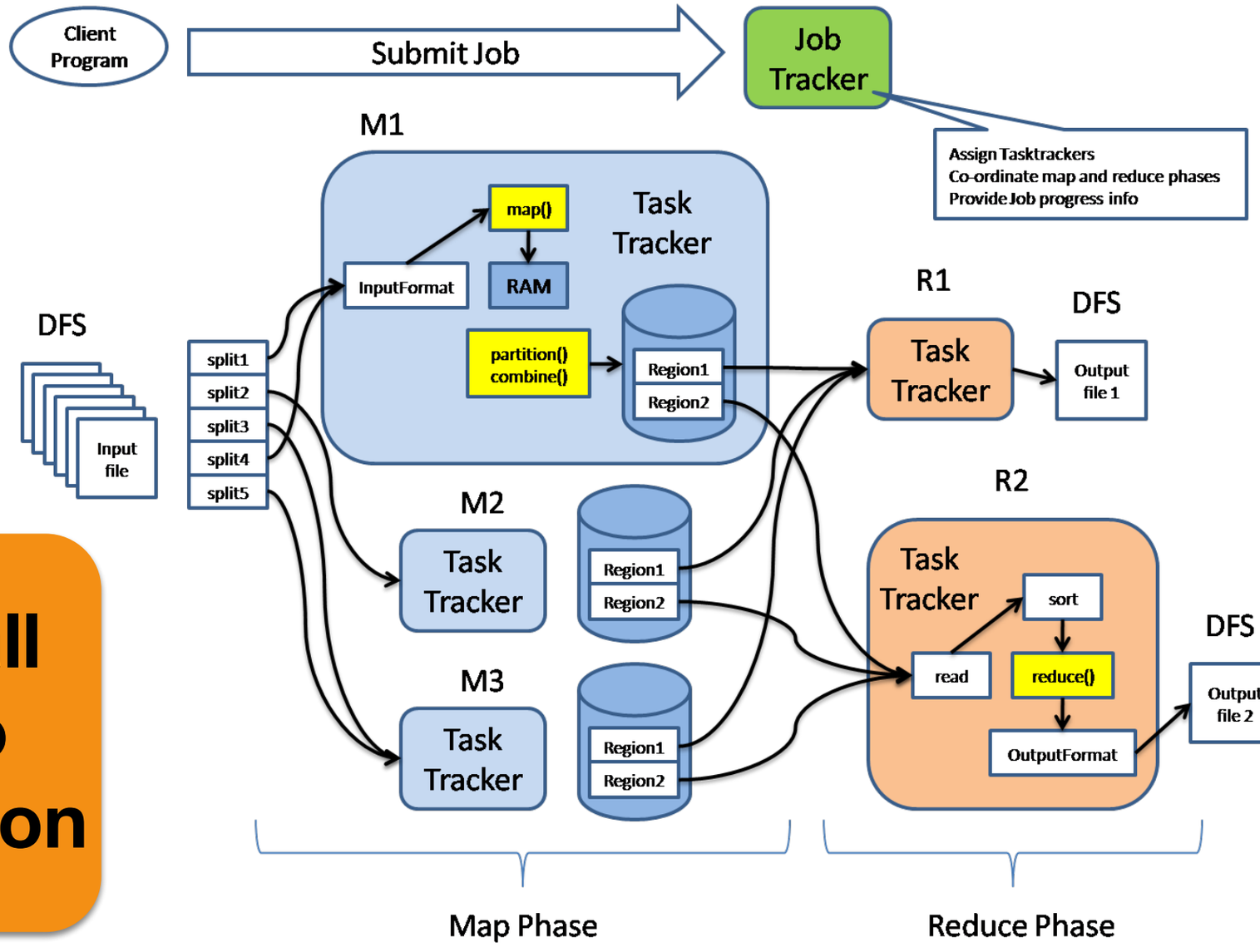
MapReduce Code - WordCount

```
public class WordCount {
   public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> { … }
   public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> { … }
   public static void main(String[] args) throws Exception {
     Configuration conf = new Configuration();
     Job job = new Job(conf, "wordcount");
     job.setOutputKeyClass(Text.class);
     job.setOutputValueClass(IntWritable.class);
     job.setMapperClass(Map.class);
     job.setReducerClass(Reduce.class);
     job.setInputFormatClass(TextInputFormat.class);
     job.setOutputFormatClass(TextOutputFormat.class);
     FileInputFormat.addInputPath(job, new Path(args[0]));
     FileOutputFormat.setOutputPath(job, new Path(args[1]));
     job.waitForCompletion(true);
   }
}
```

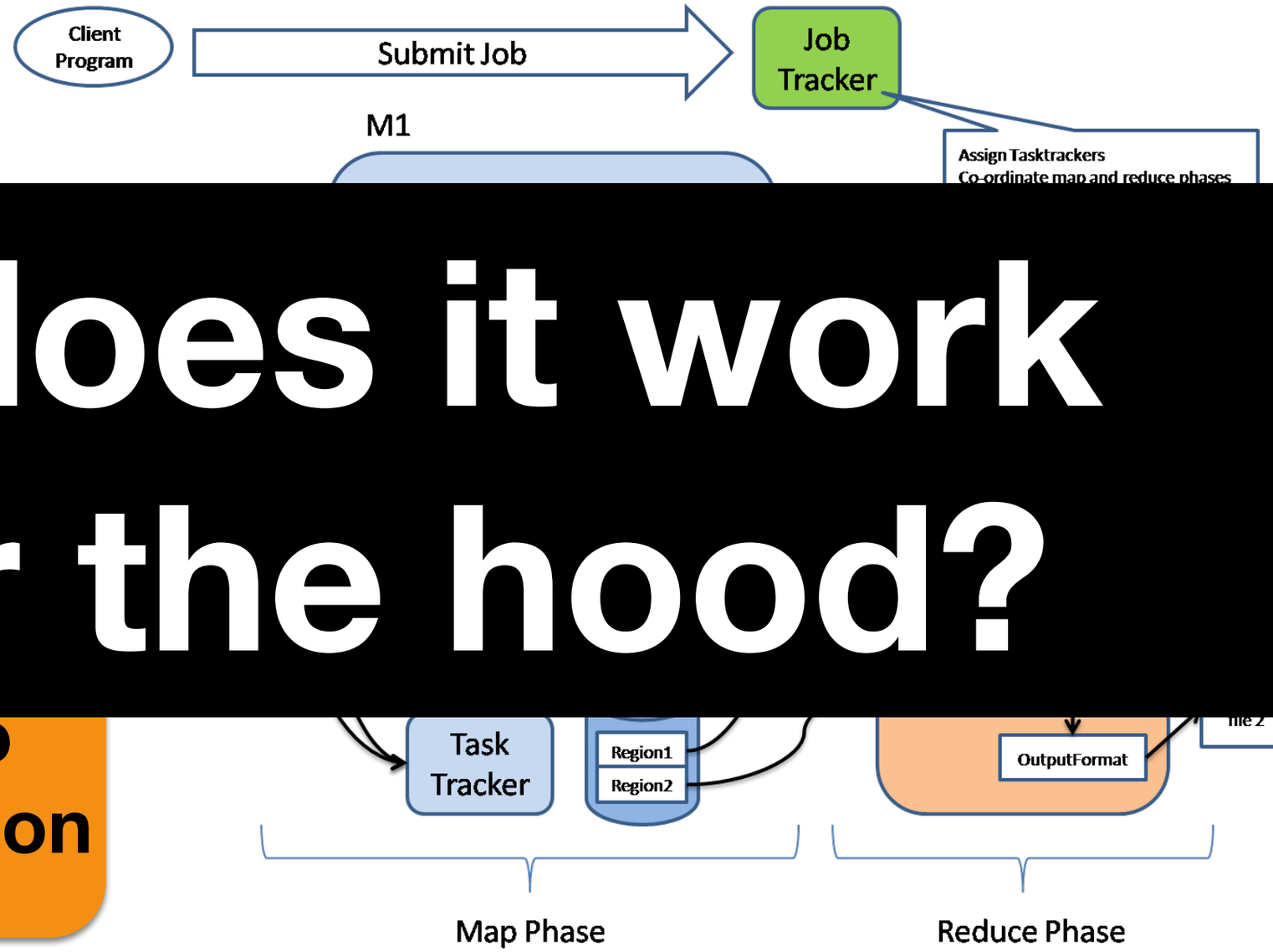
Simplified Distributed Execution

to simplify

Hide from the user all problems related to distributed computation



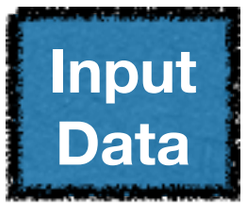
Simplified Distributed Execution



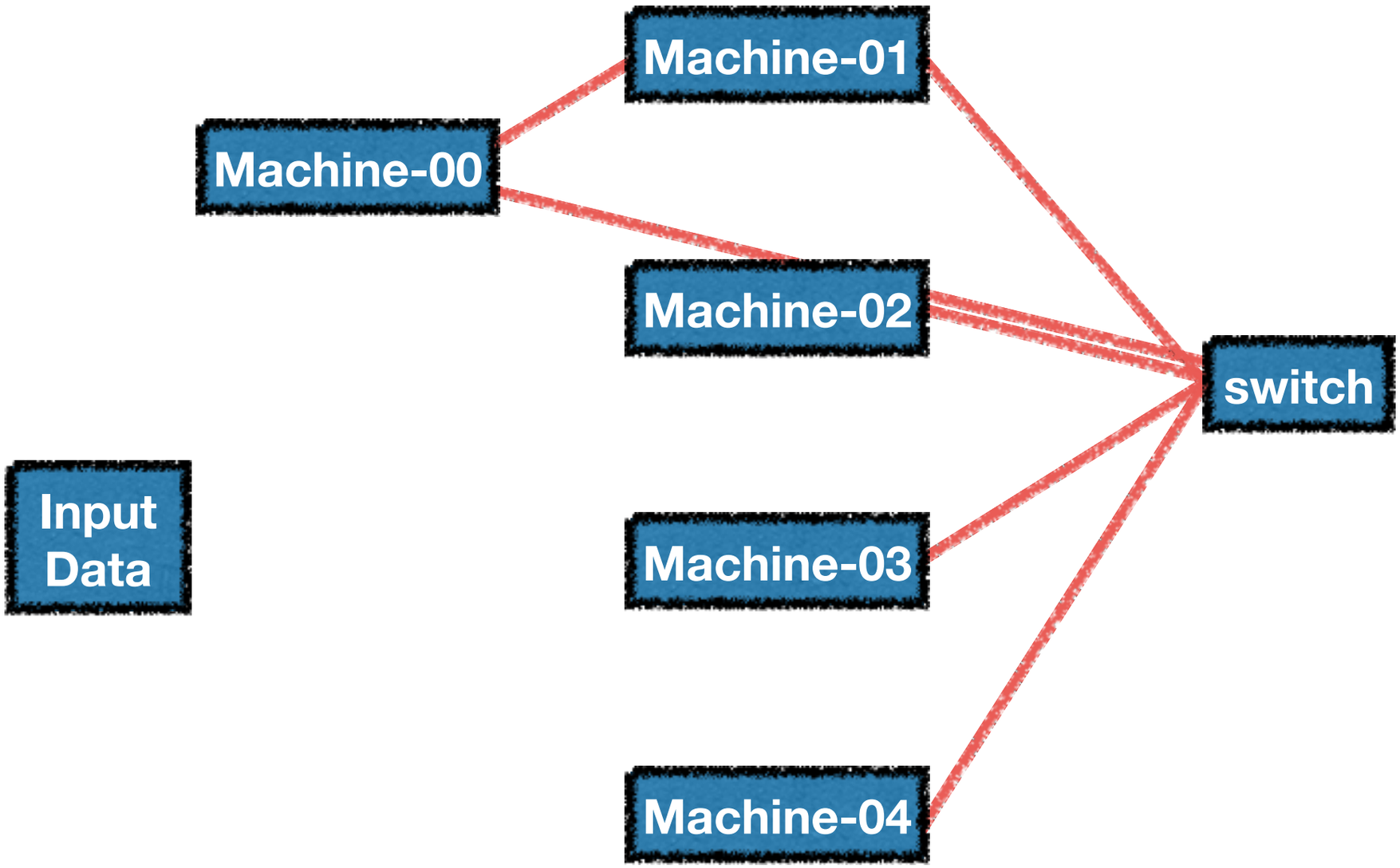
How does it work under the hood?

problems related to distributed computation

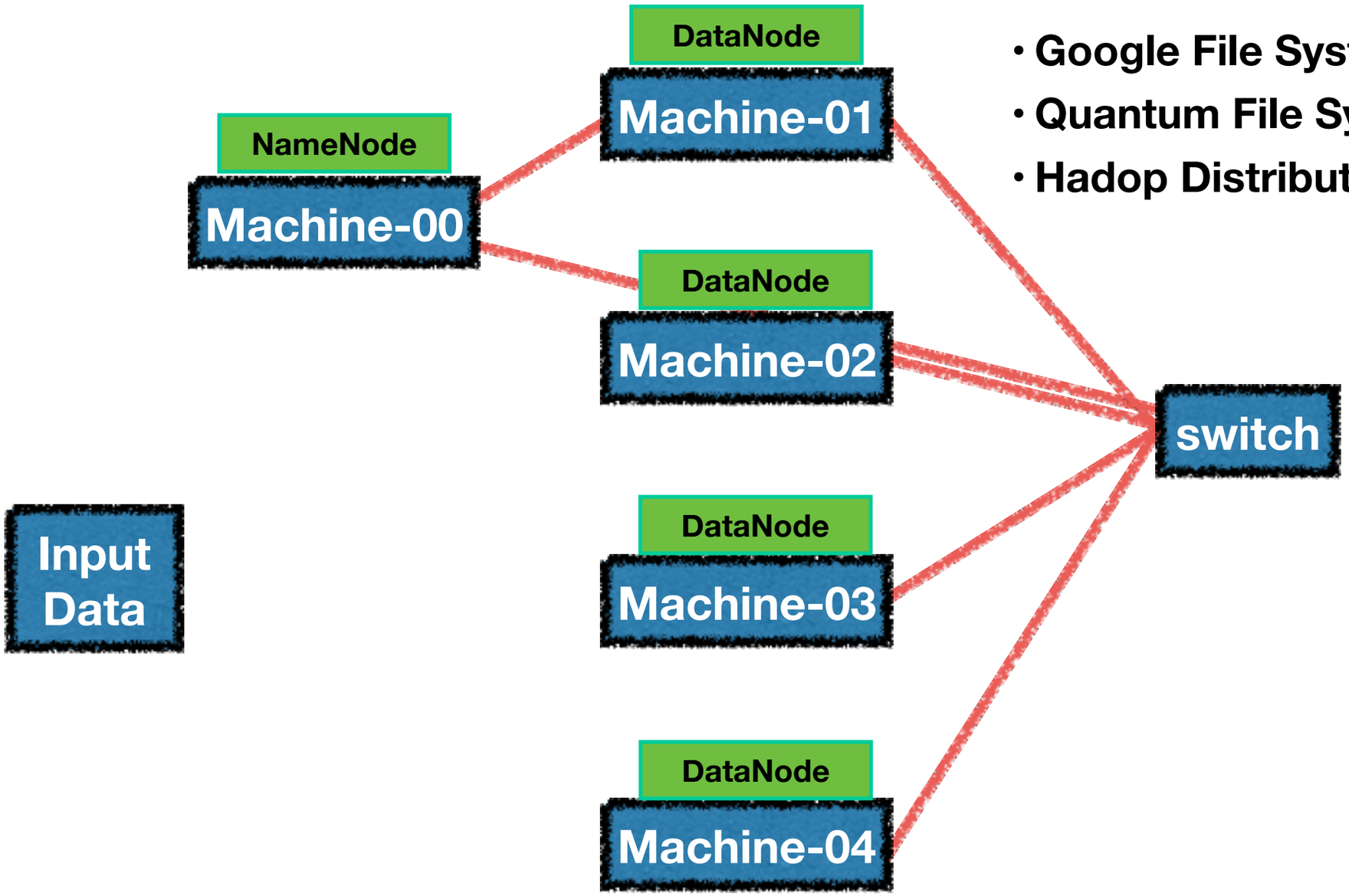
We have data…



...we have machines,

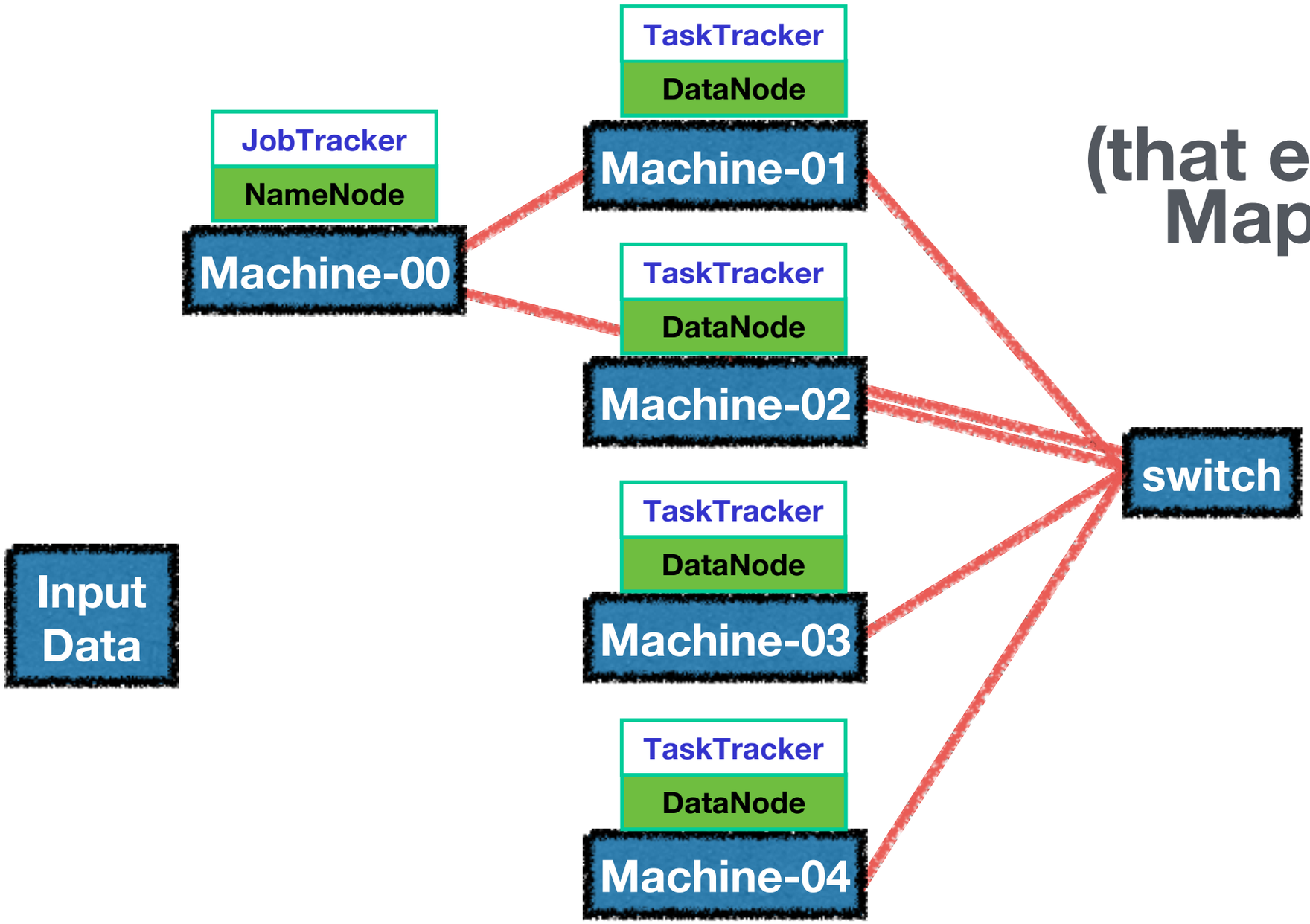


...we have processes for storing data,

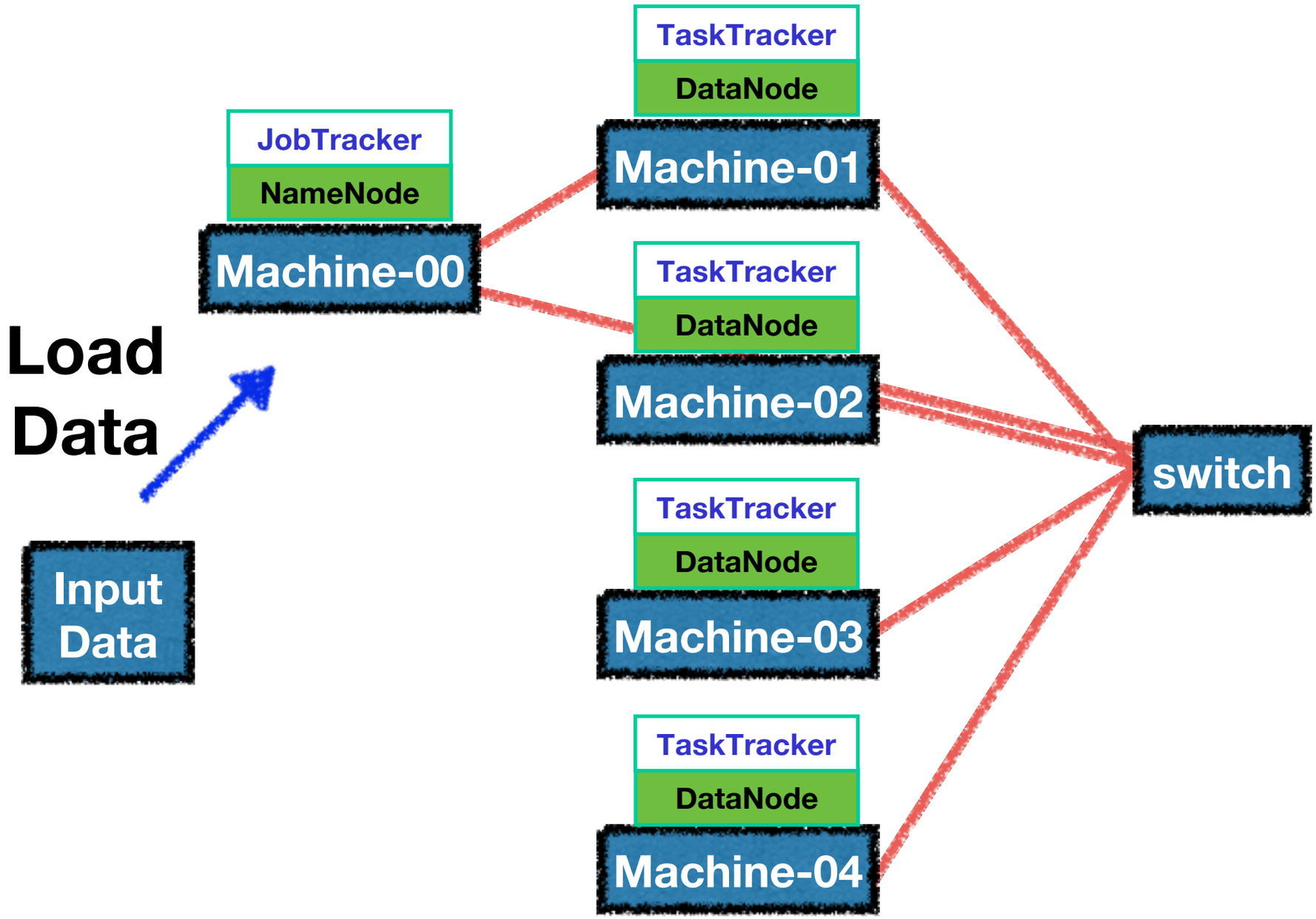


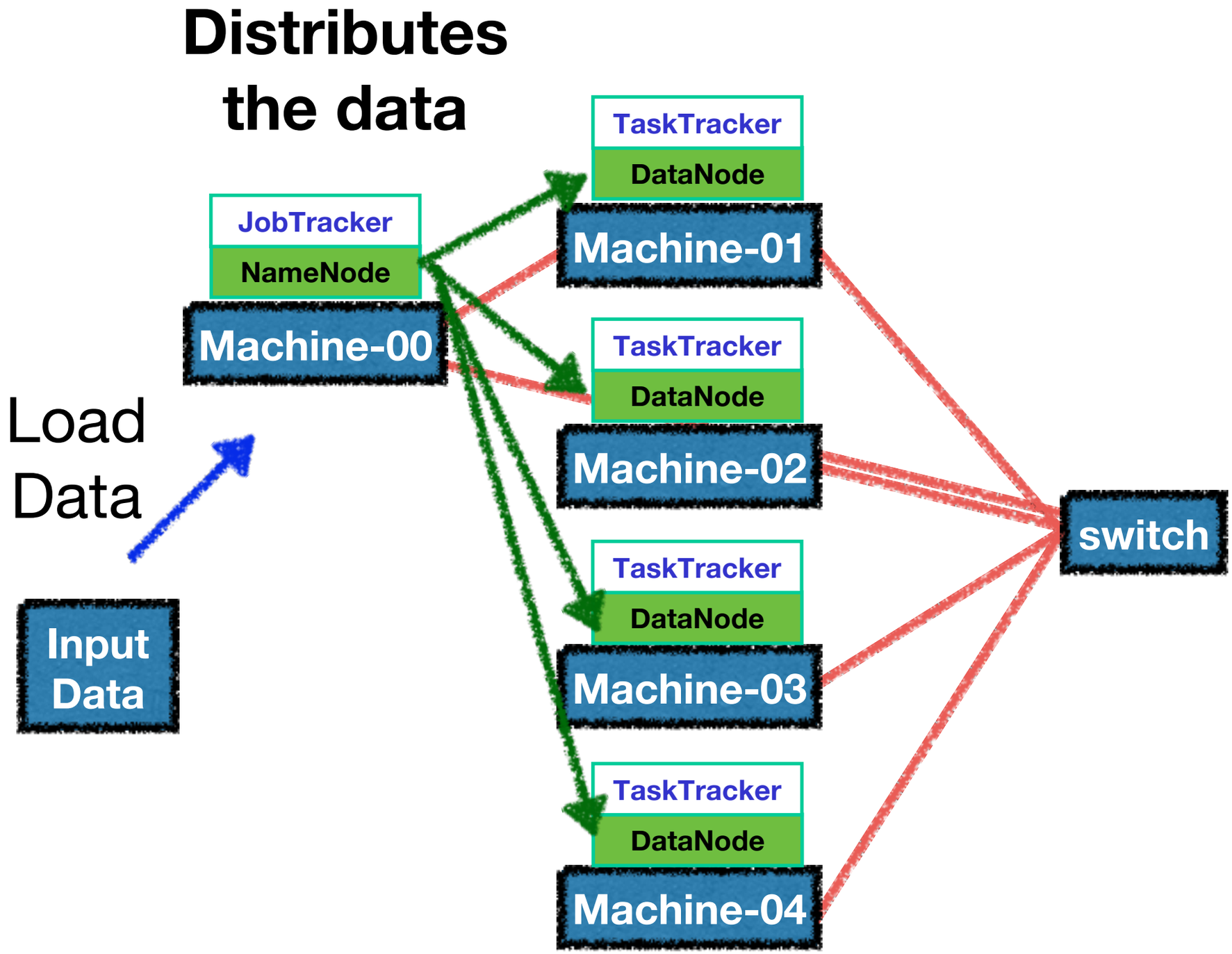
- Google File System (GFS)
- Quantum File System (QFS)
- Hadop Distributed File System (HDFS)

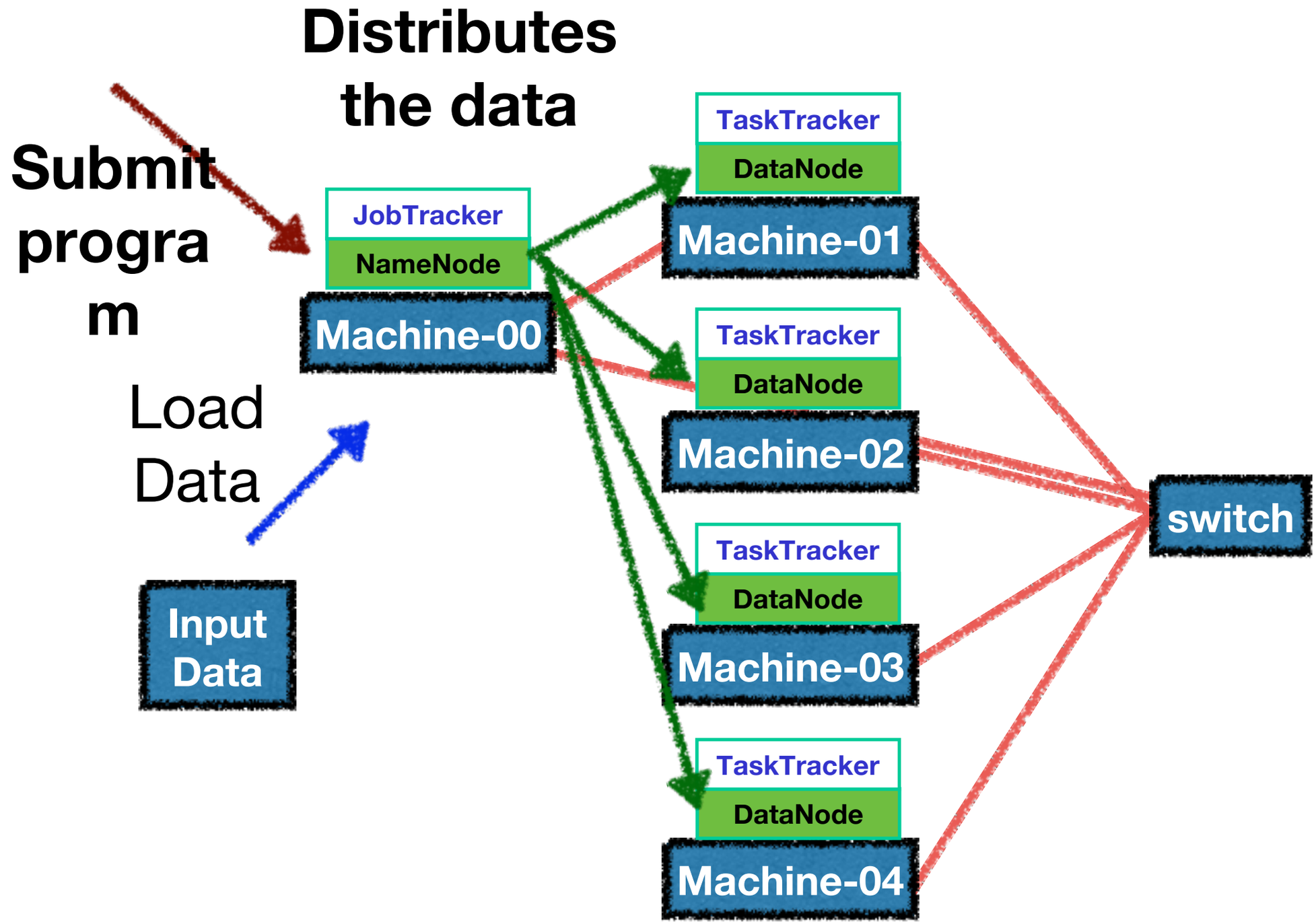
...and processes for processing data,

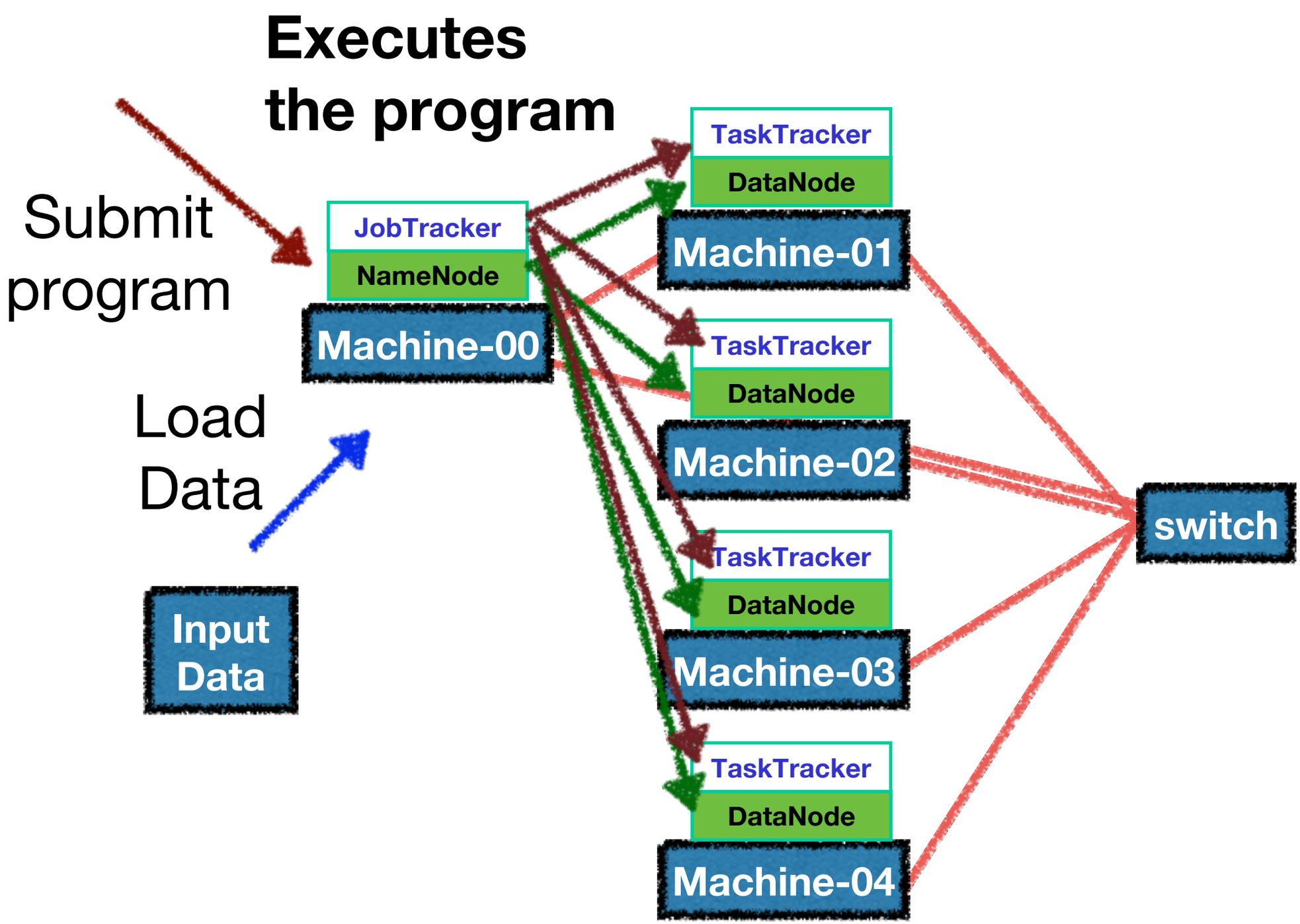


(that execute the Map & Reduce functions)


Load Data
Input Data
JobTracker
NameNode
Machine-00
TaskTracker
DataNode
Machine-01
TaskTracker
DataNode
Machine-02
TaskTracker
DataNode
Machine-03
TaskTracker
DataNode
Machine-04
switch


Distributes the data
Load Data
Input Data
JobTracker
NameNode
Machine-00
TaskTracker
DataNode
Machine-01
TaskTracker
DataNode
Machine-02
TaskTracker
DataNode
Machine-03
TaskTracker
DataNode
Machine-04
switch


Distributes the data
Submit program
Load Data
Input Data
JobTracker
NameNode
Machine-00
TaskTracker
DataNode
Machine-01
TaskTracker
DataNode
Machine-02
TaskTracker
DataNode
Machine-03
TaskTracker
DataNode
Machine-04
switch

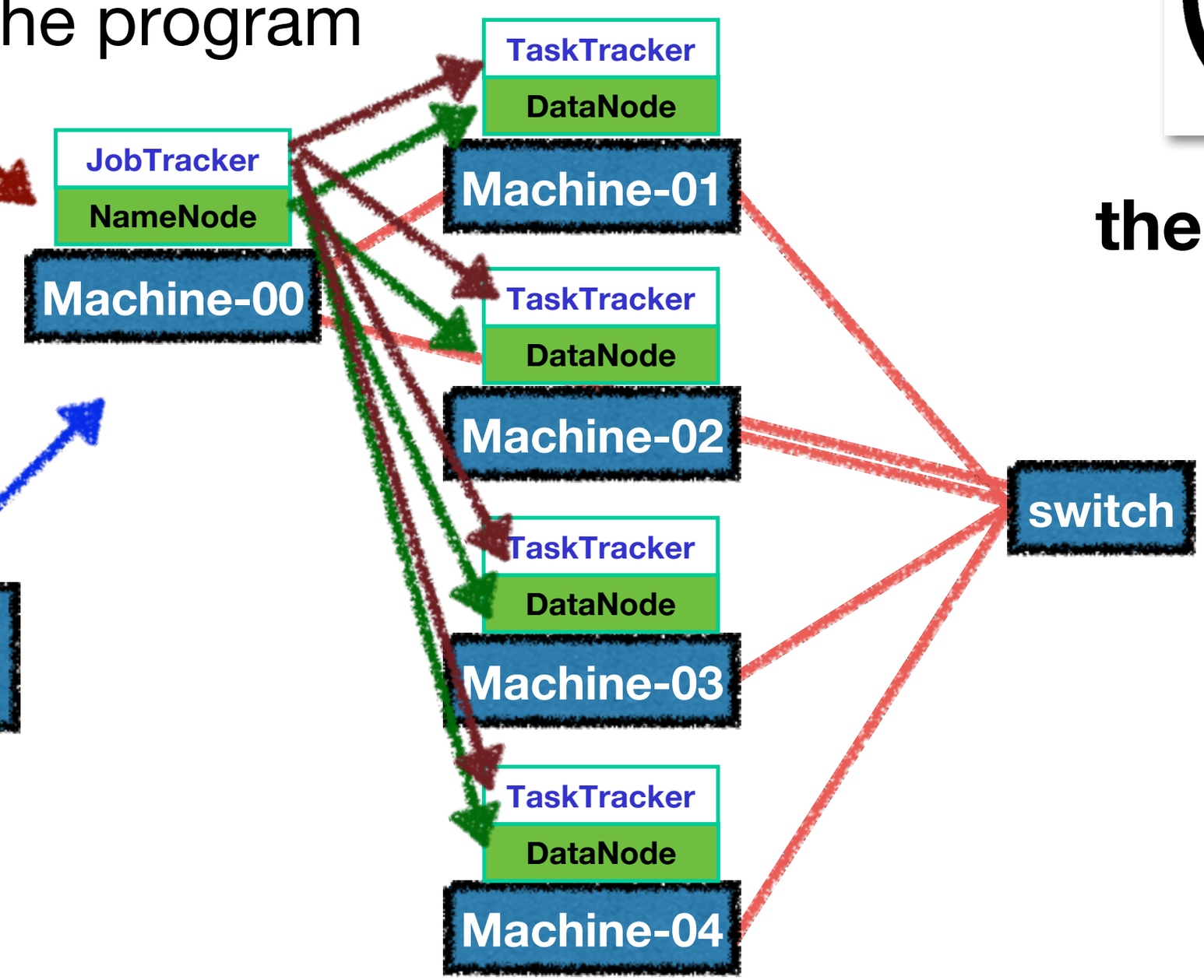

Executes
the program
Submit
program
Load
Data
Input
Data
JobTracker
NameNode
Machine-00
TaskTracker
DataNode
Machine-01
TaskTracker
DataNode
Machine-02
TaskTracker
DataNode
Machine-03
TaskTracker
DataNode
Machine-04
switch

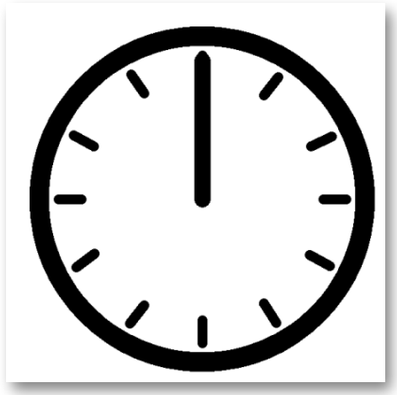
Executes
the program

Submit
program

Load
Data

Input
Data



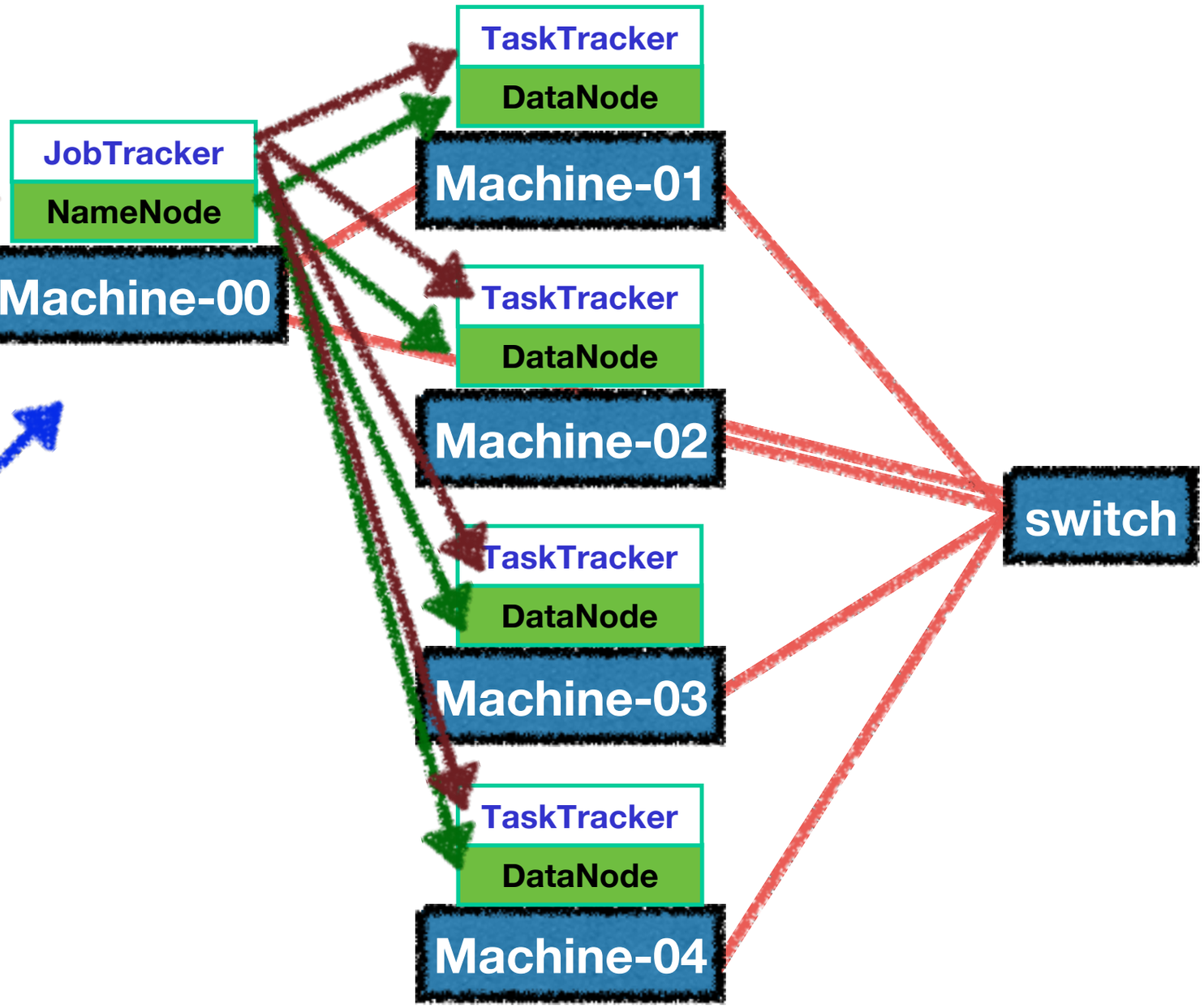


then we wait! :)

...and get results

Load Data

Input Data

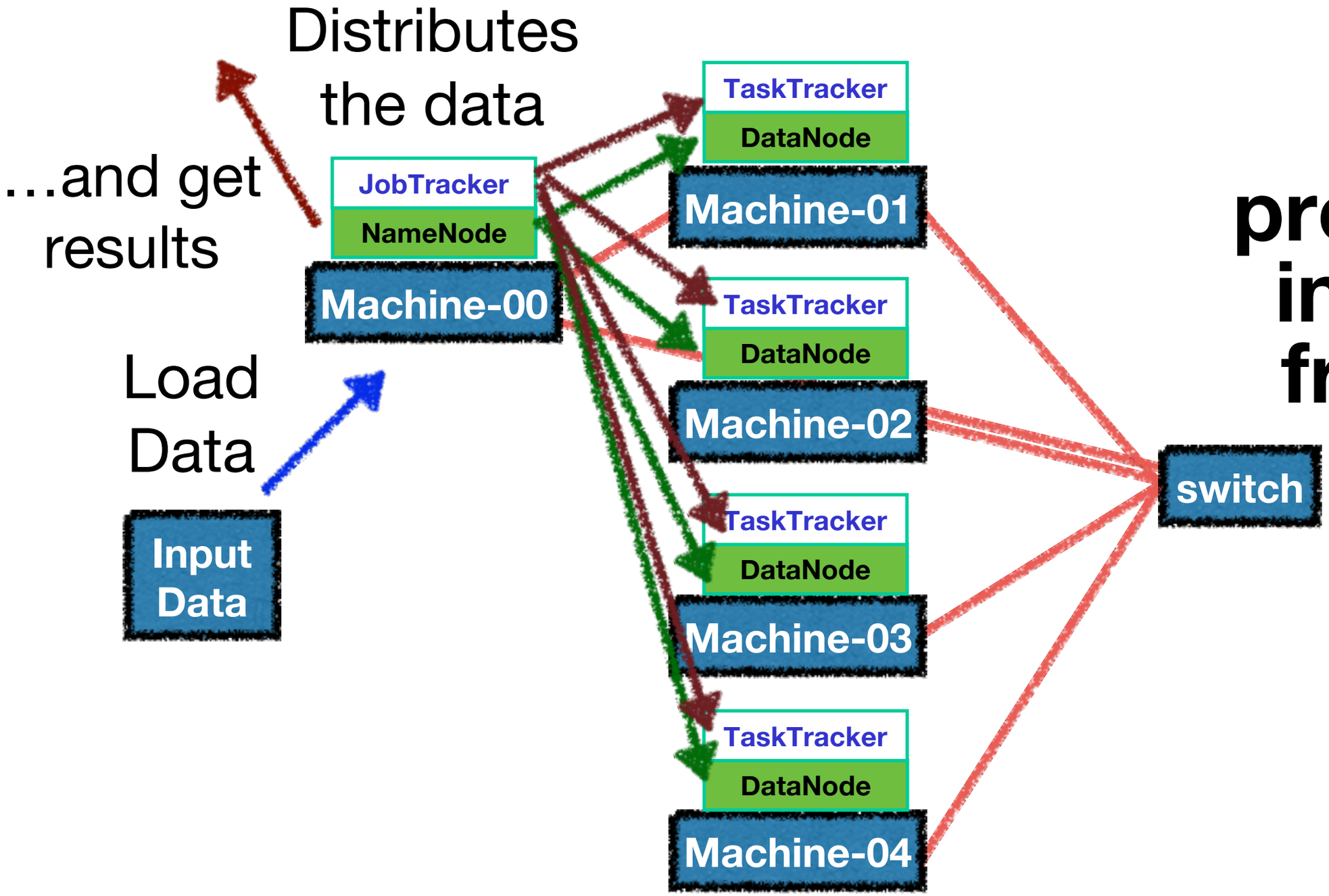


MapReduce was designed for batch processing

Not interactive or streaming processing !

How does MapReduce Scale ?

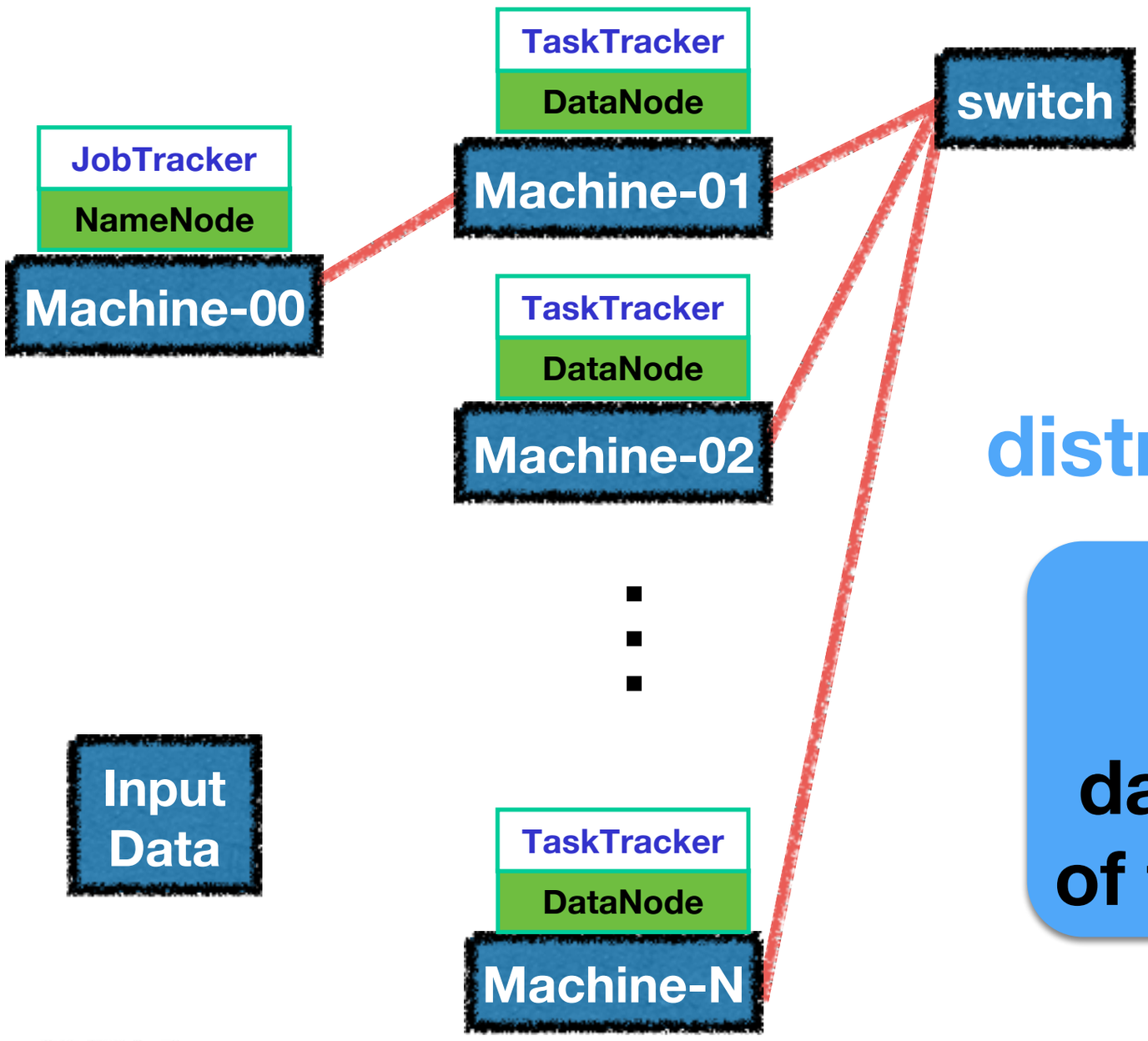
How does it Scale?



processing input data from 1 Gb to 1 Tb or 1 Pb or 1 Zb ....

How does it scale?

Just add more (cheap) machines

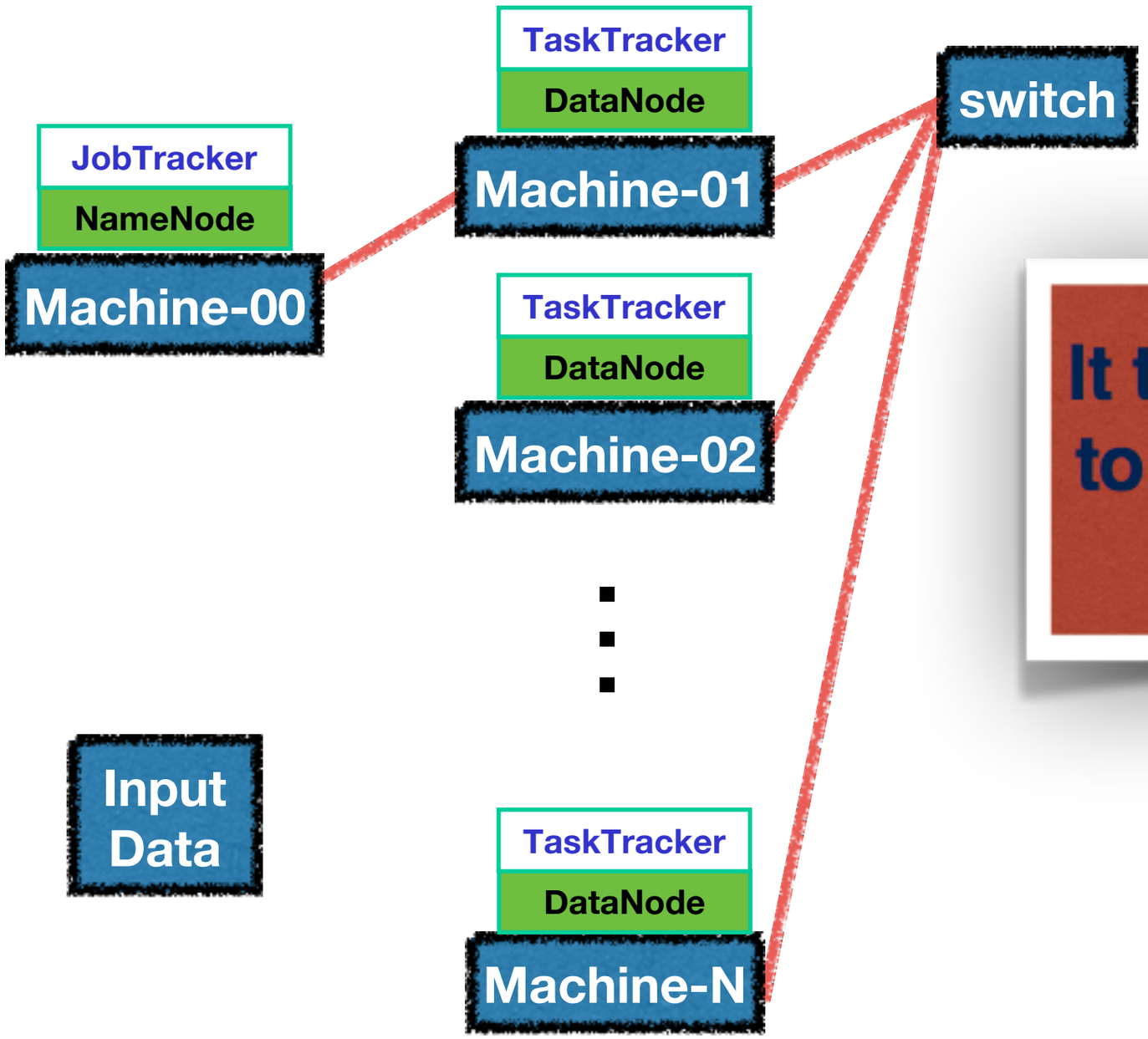


distributed data processing

Allows distributed processing of large data sets across clusters of thousands of machines

How does it scale?

Just add more (cheap) machines



It takes the computation to the data, not the data to the computation.

How do we build DBMS on top of MapReduce ?

How do we build databases on top of MapReduce?

How do we build databases on top of MapReduce?

```
SELECT name FROM
people WHERE age = '25';
```

```
José,22
Mathew,25
Mirella,24
Letícia,24
Letícia,25
```

How do we build databases on top of MapReduce?

```
SELECT name FROM
people WHERE age = '25';
```

```
José,22
Mathew,25
Mirella,24
Letícia,24
Letícia,25
```

May be any kind of unstructured data

How do we build databases on top of MapReduce?

SELECT name FROM people WHERE age = '25';

```
function map(String name,
             String document):
  // name: document name
  // document: document contents
  array = document.split(',')
  if (array[1] == '25')
      emit (array[0], true)
  else
      emit (array[0], false)
```

José,22
Mathew,25
Mirella,24
Letícia,24
Letícia,25

How do we build databases on top of MapReduce?

```
SELECT name FROM
people WHERE age = '25';
```

```
function map(String name,
             String document):
  // name: document name
  // document: document contents
  array = document.split(',')
  if (array[1] == '25')
      emit (array[0], true)
  else
      emit (array[0], false)
```

```
José,false
Mathew,true
Mirella,false
Letícia,false
Letícia,true
```

How do we build databases on top of MapReduce?

SELECT name FROM people WHERE age = '25';

```
function reduce(String word,
                Iterator isXOld):
// word: a word
// idXOld: list of ages that are 25
for each age in idXOld:
    if (age ==  true)
        emit (word)
```

José,false
Mathew,true
Mirella,false
Letícia,false
Letícia,true

How do we build databases on top of MapReduce?

```
SELECT name FROM
people WHERE age = '25';
```

```
José,false
Mathew,true
Mirella,false
Letícia,false
Letícia,true
```

How do we build databases on top of MapReduce?

```
SELECT name FROM
people WHERE age = '25';
```

- We can use just Map function
- We can perform many kinds of computation (e.g., Graph, Machine Learning, SQL)

Readings

- **[paper]** Dean, Jeffrey and Ghemawat, Sanjay. MapReduce: Simplified Data Processing on Large Clusters (OSDI'04). Material at: http://dx.doi.org/10.1145/1327452.1327492
- **[implementation]** The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing. http://hadoop.apache.org/

Questions?

Eduardo Cunha de Almeida <eduardo@inf.ufpr.br>
Edson Ramiro Lucas Filho <edson.lucas@uni.lu>