MR - A MapReduce Programming Language

W4115 Programming Language & Translator
Fall 2011

Siyang Dai
Zhi Zhang
Zeyang Yu
Jinxiong Tan
Shuai Yuan

Motivation

```
class HelloWorld {
    public static void main ( String argv[] ) {
        System.out.println("Hello World!");
    }
}
```

All we need actually is one line:

```
    System.out.println("Hello World!");
```

Motivation

Word Count

Helloworld for Mapreduce

```java
import java.io.IOException;
import java.lang.InterruptedException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {
    /**
     * The map class of WordCount.
     */
    public static class TokenCounterMapper extends
            Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    /**
     * The reducer class of WordCount
     */
    public static class TokenCounterReducer extends
            Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values,
                Context context) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable value : values) {
                sum += value.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }

    /**
     * The main entry point.
     */
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args)
                .getRemainingArgs();
        Job job = new Job(conf, "Example Hadoop 0.20.1 WordCount");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenCounterMapper.class);
        job.setReducerClass(TokenCounterReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Motivation

Why not like this:

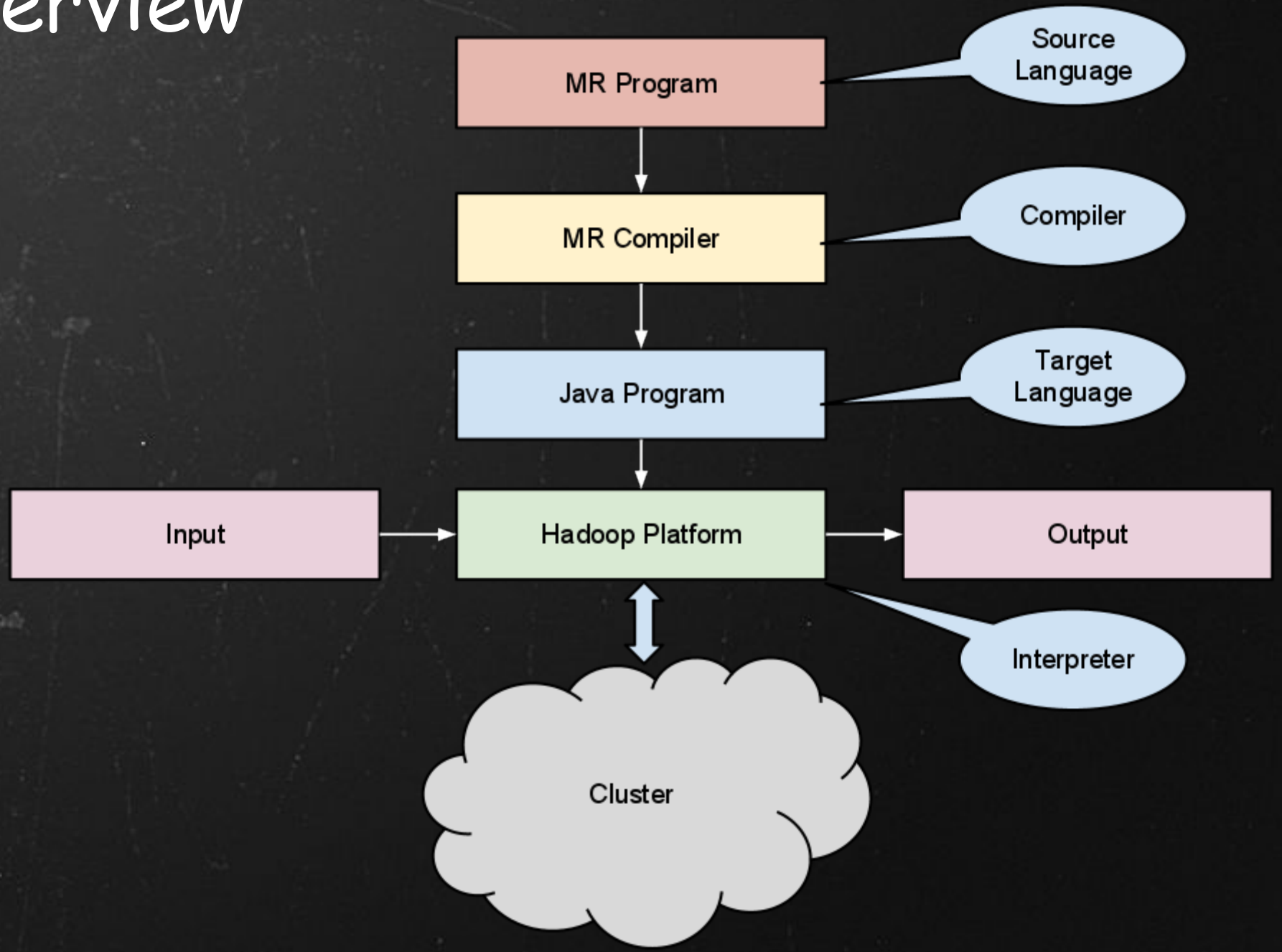
```
#JobName = "WordCount"

def wordcount_map(offset, line) <(Int, Text) -> (Text , Int)> : Mapper
{
    List<Text> words = split line by " ";
    foreach word in words
        emit(word, 1);
}

def wordcount_reduce (word, counts) <(Text , Int) -> (Text, Int)> : Reducer
{
    Int total = 0;
    foreach count in counts
        total = total + count;

    emit(word, total);
}
```

Overview


MR Program
Source Language
MR Compiler
Compiler
Java Program
Target Language
Input
Hadoop Platform
Output
Interpreter
Cluster

Tutorial: wordcount.mr

```
#JobName = "WordCount"
//map function definition
def wordcount_map <(Int, Text) -> (Text , Int)> (offset, line): Mapper
{
List<Text> words;
      words = split line by " ";
foreach Text word in words {
emit(word,  1);
}
}
//reduce function definition
def wordcount_reduce <(Text , Int) -> (Text, Int)> (word, counts): Reducer
{
Int total;
total = 0;
foreach Int count in counts {
total = total + count;
}
emit(word, total);
}
```

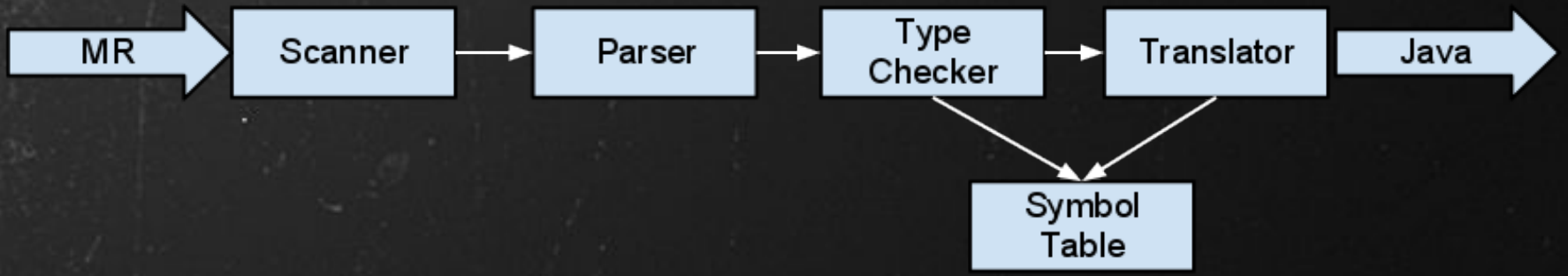
Map function

```
//map function definition
def wordcount_map <(Int, Text) -> (Text , Int)> (offset, line): Mapper
{
List<Text> words;
words = split line by " ";
foreach Text word in words {
emit(word,  1);
}
}
```

Reduce function

```
//reduce function definition
def wordcount_reduce <(Text , Int) -> (Text, Int)> (word, counts): Reducer
{
Int total;
total = 0;
foreach Int count in counts {
total = total + count;
}
emit(word, total);
}
```

Compiler Structure



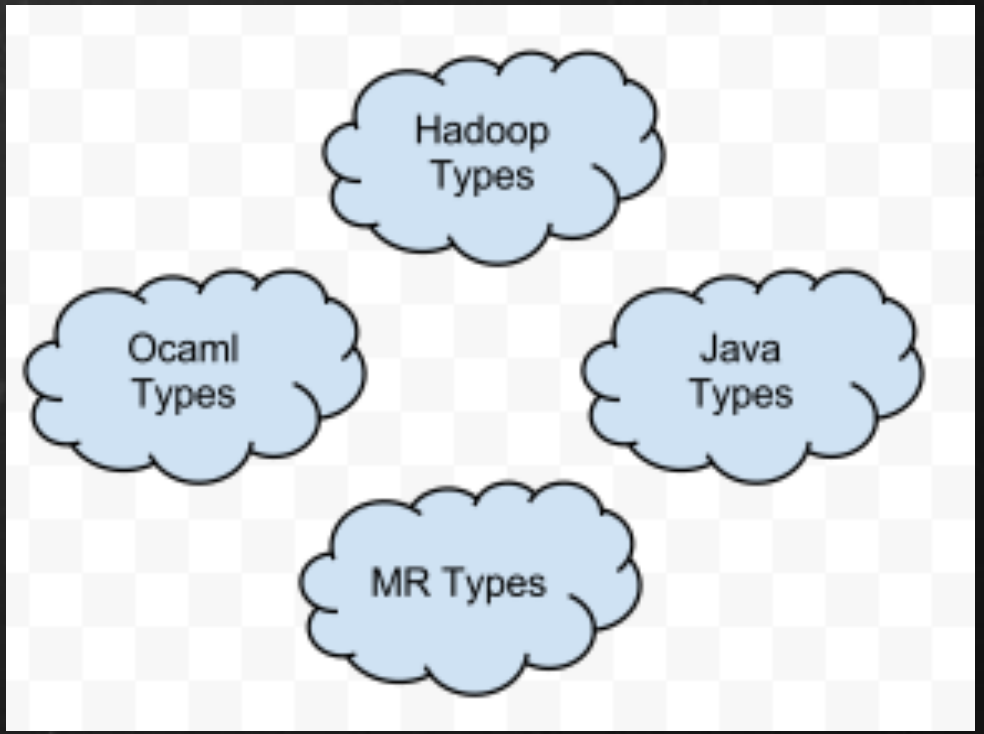
Translator

Mapper / Reducer (Parameters in Hadoop types) {
mapper / reducer logic
emit the values in Hadoop types
}

Problem:

Hadoop types do not support basic operations.

Types Puzzle



Translator

Mapper / Reducer (Parameters in Hadoop types) {

Convert from Hadoop types to Java types

mapper / reducer logic

Convert from Java types to Hadoop types

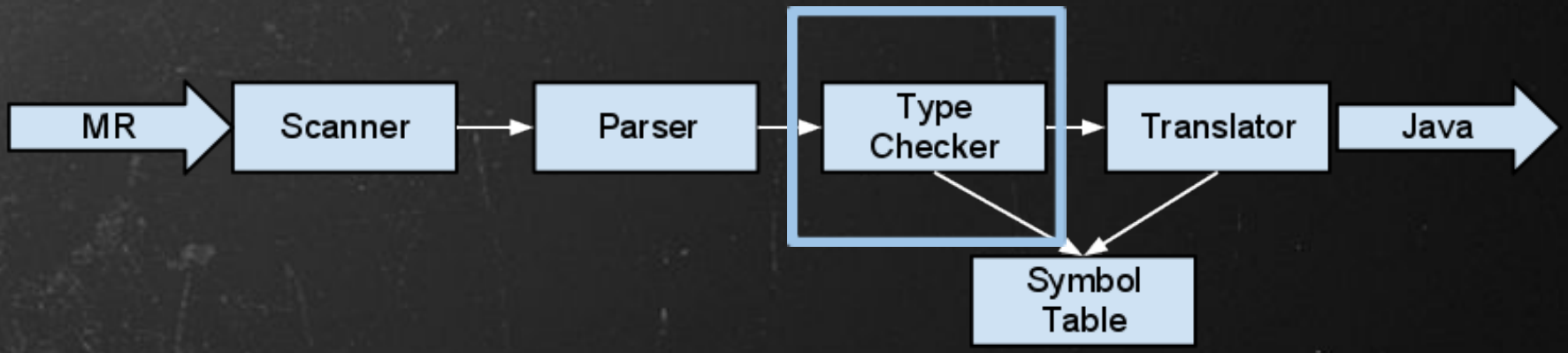
emit the values in Hadoop types

}

Solution:

Another layer of indirection

Type Checker



Prevent Ridiculous Expression/Statement

e.g.

```
Int a ="oops"; //illegal assignment

if("oops"){   //it should be boolean type
    ...
}
```

Type Checker

Variable Definition
Assignment Operation
Binary Operation (Arithmetic, Relational, Logical)
Split Expression
If-Expression
Foreach Statement
Emit Statement

Lessons Learned

1. Parametric Polymorphism
2. Think Recursively
3. Universal Solution - Indirection
4. Teamwork

Demo